INGnosis: Revista de investigación científica
Universidad Cesar Vallejo, Perú
E-ISSN 2414-8199
10 (1). Julio-Diciembre 2024
http://revistas.ucv.edu.pe/index.php/INGnosis//


# Desarrollo de Inteligencia Artificial para la detección de manzana Granny Smith en mal estado mediante software Python

## Development of Artificial Intelligence for the detection of Granny Smith apples in poor condition using Python software

## Desenvolvimento de Inteligência Artificial para detecção de maçãs Granny Smith em mau estado utilizando software Python

**Castillo Alba, Isabela** [1]
https://orcid.org/0000-0002-5157-1466
**Castro Romero, Jennifer** [2]
https://orcid.org/0000-0001-9702-0259
**Marcos Navarro, Junior** [3]
https://orcid.org/0000-0002-5316-3968
**Meza Castillo, Daniela** [4]
https://orcid.org/0000-0001-8328-7681
**Pozo Rodríguez, Aaron** [5]
https://orcid.org/0000-0002-7192-7132
**Vega Saldaña, Junior** [6]
https://orcid.org/0000-0002-8025-7820

Recibido: 15.02.2024
Aceptado: 16.04.2024

**RESUMEN**
El presente proyecto abarca la implementación de un sistema avanzado para evaluar las características de clasificación de manzanas chilenas en función de su color y perímetro. Además, es capaz de detectar manzanas en estado de descomposición. Para lograr esto, se emplea un programa de visión artificial desarrollado en Python, el cual se ejecuta en una laptop HP. El sistema analiza las características de las manzanas utilizando el historial de IDLE Shell, donde se registran momentos de captura de imagen, contornos y centroides. Esto permite determinar si una manzana es aceptada o rechazada. Nuestro algoritmo detecta posibles anomalías, gracias a un entorno controlado y a una Cámara web Jetion PJT-DCM141 con conexión USB de 720p, que genera imágenes en un entorno estable e iluminado. Con el uso de nuestra programación, hemos alcanzado una eficiencia del 93% y una presión del 95%, lo que consideramos resultados satisfactorios para nuestra investigación.
***Palabras clave**:* Sistema avanzando, Visión Artificial, Historial de IDLE Shell, programación Python.

**ABSTRACT**
This project covers the implementation of an advanced system to evaluate the classification characteristics of Chilean apples based on their color and perimeter. In addition, it is capable of detecting apples in a state of decomposition. To achieve this, a computer vision program developed in Python is used, which runs on an HP laptop. The system analyzes the characteristics of the apples using the IDLE Shell history, where image capture

[1] Universidad Cesar Vallejo. Chimbote. Perú. Bachiller. castilloalbaisabela24@gmail.com
[2] Universidad Cesar Vallejo. Chimbote. Perú. Bachiller. askyfullofstars_15@hotmail.com
[3] Universidad Cesar Vallejo. Chimbote. Perú. Bachiller. marcosnabraham@gmail.com
[4] Universidad Cesar Vallejo. Chimbote. Perú. Bachiller. danielamezacastillo14@gmail.com
[5] Universidad Cesar Vallejo. Chimbote. Perú. Bachiller. apozorod@gmail.com
[6] Universidad Cesar Vallejo. Chimbote. Perú. Bachiller. kidbourbon1@gmail.com

moments, contours and centroids are recorded. This allows you to determine whether an apple is accepted or rejected. Our algorithm detects possible anomalies, thanks to a controlled environment and a Jetion PJT-DCM141 Webcam with 720p USB connection, which generates images in a stable and illuminated environment. Using our programming, we have achieved an efficiency of [93%] and a pressure of [95%], which we consider satisfactory results for our research.
***Keywords**:* System Advancing, Computer Vision, IDLE Shell History, Python Programming.

**RESUMO**
Este projeto abrange a implementação de um sistema avançado para avaliar as características de classificação das maçãs chilenas com base na sua cor e perímetro. Além disso, é capaz de detectar maçãs em estado de decomposição. Para isso, é utilizado um programa de visão computacional desenvolvido em Python, que roda em um laptop HP. O sistema analisa as características das maçãs utilizando o histórico IDLE Shell, onde são registrados os momentos de captura da imagem, contornos e centróides. Isso permite determinar se uma maçã é aceita ou rejeitada. Nosso algoritmo detecta possíveis anomalias, graças a um ambiente controlado e uma Webcam Jetion PJT-DCM141 com conexão USB 720p, que gera imagens em ambiente estável e iluminado. Utilizando nossa programação, alcançamos 93% de eficiência e 95% de pressão, resultados que consideramos satisfatórios para nossas pesquisas.
***Palavras-chave:*** Avanço de sistemas, Visão Computacional, História do IDLE Shell, Programação Python.

## Introducción

Desde hace años, como peruanos nos sentimos orgullosos de importar una gran variedad de manzanas por millones de dólares. Entre las más populares en nuestro grupo se encuentra la Granny Smith. Además, es importante saber cómo analizar si estas manzanas están en mal estado (MAPA, 2020), dado que la industria frutícola en la economía peruana está decayendo en su participación en el mercado internacional (INEI, 2023), podemos notar las causas importantes donde las exportaciones cayeron un 0.1% entre los principales problemas los fenómenos climáticos (ComexPerú, 2023), y no sabemos si las manzanas son de la misma calidad a nuestros mercados (PlazaVea, 2024), como dato resalta la importancia crítica de evaluar minuciosamente el estado de las manzanas en su afectación al cambio climático (Sepúlveda, Araya, 2017), donde estudios de análisis de riesgo generado por SENASA quieren evitar plagas de moscas en las frutas y no sabemos si a la fecha los problemas han sido solucionados (#SomosNoticia, 2021), estas frutas pueden terminar en mercados mayoristas, poniendo en riesgo la salud de numerosos consumidores, es alarmante descubrir que decomisan grandes cantidades de frutas en estado de deterioro (La República, 2019).

Este enfoque no solo consolida la reputación del Perú como un proveedor confiable de frutas frescas, sino que también impulsa la aplicación de inteligencia artificial para el uso en empresas, mercados mayoristas y en personas interesadas en la aplicación de este lenguaje de programación para una mejor calidad de nutrición (La República, 2019), entonces nos apoyamos de autores para ayudarnos a realizar esta investigación, en nivel internacional, En Honduras, Serrano Fuente, Lizardo Zelaya y Ordoñez Avila (2020) desarrollaron un sistema de visión artificial y redes neuronales para analizar granos de café. Este sistema, entrenado con 196 imágenes, alcanzó una eficiencia del 97.6%, ayudando a los productores a mejorar la calidad y acelerar el proceso de selección de granos. (IngiVision, 2021), En otras investigaciones por Ohali desarrolló un sistema de clasificación de frutas utilizando visión por computadora. Su proyecto se centró en los dátiles, utilizando imágenes RGB y evaluando tres características de calidad. Logró una precisión del 80% en la clasificación. (Al Ohali, 2010), En Ecuador, Rivera Carla desarrolló un sistema inteligente de detección de estrés hídrico para evaluar la calidad de cultivos de lechuga. Utilizando visión artificial y redes neuronales, logró un índice de error del 3.58% y una precisión del 93.9% en su detección. (Rivera, 2023). En México, Bravo José Luis concluyó su proyecto de detección automática del daño causado por el gusano cogollero en plantas de maíz. Utilizó procesamiento digital de imágenes para segmentar y clasificar las hojas en los cultivos de maíz en campo, logrando un 82.6% de eficiencia (Bravo, 2020).

Esto nos proporciona una idea especifica en cómo se puede detectar manzanas en mal estado, pero nos apoyaremos con ideas a nivel nacional para mejorar nuestra investigación, para Huillca desarrolló un

sistema de clasificación para aguaymanto y fresas en la región de Huánuco, usando redes neuronales convolucionales. Al usar bibliotecas como Numpy, TensorFlow y OpenCV, logró una precisión del 91.33% para aguaymanto y del 90.5% para fresas, con tiempos de respuesta de 81 ms y 117 ms, respectivamente (Huillca, 2022). En otra investigación, Burgos Zavaleta, Ulloa Baquedano, Aguilar Acevedo, Robles Malqui y Julián Suarez (2022), crearon un sistema de visión artificial para detectar abolladuras en latas de atún en Trujillo. Realizaron cinco series de pruebas con diez latas cada una, utilizando un programa Python para simular el funcionamiento de la máquina. Lograron una eficiencia del 90.4%, lo que es muy beneficioso para la empresa en términos de producción y calidad (Burgos Zavaleta et al., 2022, 2022). Cueva Caro empleó redes neuronales para evaluar la acidez y los grados Brix de la Pitahaya Amarilla en la Amazonía. Al analizar 12 muestras por nivel de madurez, logró una precisión del 100%, alcanzó un tiempo promedio de 0.8 segundos por imagen en el proceso (Cueva, 2022). En culminación para el autor Ryan Abraham y Jair Milton desarrollaron un Sistema de Visión Artificial para inspeccionar fechas en productos industriales alimenticios. Utilizando Python, crearon un sistema que mejoró la calidad de los productos con una eficiencia del 97.2% (León y García, 2018).

Este proyecto tiene como finalidad mejorar la calidad del producto final al detectar posibles imperfecciones en las manzanas antes de que estas lleguen al consumidor. Esta medida no solo tiene el potencial de elevar la satisfacción del cliente al recibir productos de mayor calidad, sino que también puede fortalecer su lealtad hacia la marca al demostrar un compromiso con la excelencia y la atención al detalle teniendo como objetivo general, ‘‘Desarrollar una inteligencia artificial para detectar manzanas Granny Smith en mal estado’’. Con ello, nuestro primer objetivo específico, ‘‘Implementar un ambiente controlado para la obtención de imagen para validar a futuro nuestra eficiencia y precisión’’. Con ello, nuestro segundo objetivo específico ‘‘Entrenar un modelo de IA en Python mediante algoritmos de machine learning para detectar manzanas con anomalías’’.

**Material y métodos:**

Diseño captura de imágenes, se creará un ambiente controlado mediante una caja, cuyo interior estará completamente revestido en color blanco. Este diseño facilitará la detección precisa de la manzana al capturar las fotografías, ya que el fondo blanco proporcionará un contraste óptimo. Además, la caja estará equipada con iluminación LED de alta calidad. Las luces LED ofrecerán una iluminación directa y uniforme, eliminando sombras indeseadas y resaltando todos los detalles de la manzana (Martin Monroy, 2006). Este entorno controlado garantizará condiciones ideales para la captura de imágenes precisas.

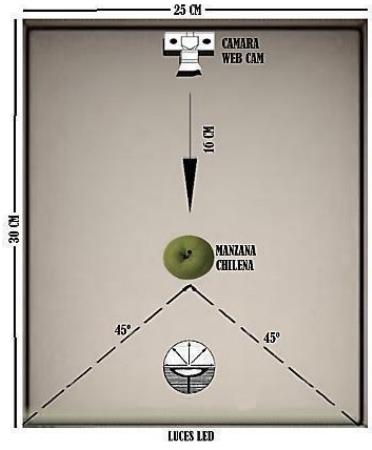


***Figura 1.*** Ambiente controlado para capturas imágenes de manzanas chilenas.

En la Figura 1, podemos observar que la luminosidad de las luces LED se dirige directamente en un ángulo de 45°, en el extremo opuesto, una cámara web captura las manzanas deseadas desde una distancia de 16 cm.

Biblioteca y adquisición de frame, es importante implementar bibliotecas para compensar a realizar la programación tanta biblioteca NumPy y OpenCV, ofrecen imágenes en operaciones de visión por computadora de manera rápida y eficaz (DataScientest, 2019).

**Tabla 1.**
*Comando importación biblioteca.*

| ***BIBLIOTECA ‘‘np & cv2’’*** | ***DESCRIPCION*** |
|---|---|
| import numpy as np<br>import cv2 | Este comando importa NumPy como 'np' y OpenCV como 'cv2' en Python, permite el procesamiento eficiente de imágenes y visión por computadora. |

En la Tabla 1, el código al establecer alias para NumPy y OpenCV, dos herramientas clave para el desarrollo de aplicaciones de IA y visión por computadora en Python. Asimismo, en la tabla 2 describe el inicio de captura de video, donde al presionar la letra ‘‘z’’ se captura un frame, representando la imagen capturada en ese momento. Tener en cuenta que, si modificamos el parámetro 'VideoCapture' de 0 a 1, se captura un video externo del computador en lugar de la cámara integrada.

**Tabla 2.**
*Comando captura imagen de manzana chilena*

| ***CAPTURA DE IMAGEN ‘‘z’’*** | ***DESCRIPCION*** |
|---|---|
| cap = cv2.VideoCapture(0) while (True):<br>ret, frame = cap.read() cv2.imshow('MznCln', frame)<br>if cv2.waitKey(1) & 0xFF == ord('z'): break<br>cv2.imwrite('manzanachilena.jpg',fram e)<br>cap.release()<br>cv2.destroyAllWindows() | Este código en Python utiliza OpenCV para capturar video de la cámara del sistema. Muestra en tiempo real lo que la cámara ve en una ventana llamada 'Objeto' y guarda la última imagen capturada como 'manzanachilena.jpg' cuando se presiona la tecla 'z'. |

Estado de manzana identificada, debemos analizar aquellas manzanas que se encuentran en deterioro y algunas que pueden estar aptos a simple vista para el consumo humano, mientras el tiempo de investigación va pasando podemos notar gran diferencia.

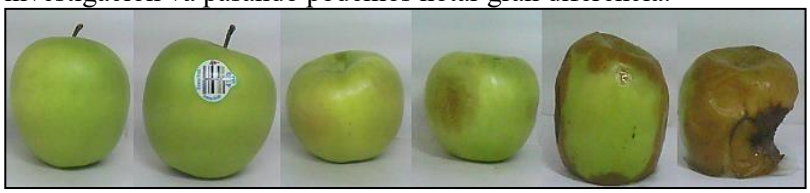

***Figura 2.*** Estado notorio de la manzana.

Captación de imagen, al momento de captura la imagen, se abrirá una ventana con el nombre ‘‘Imagen manzana chilena reducida’’, donde tenemos mucho cuidado en preservar la relación de aspecto, para no cambia la calidad de la imagen original (Martínez, 2022).

**Tabla 3.**
*Comando para reducir la dimensión de imagen*

| ***REDIMENSION DE PIXELES*** | ***DESCRIPCION*** |
|---|---|
| img=cv2.imread('manzanachilena.jpg')<br>redim=cv2.resize(img,(400,350)) cv2.imshow('imagen manzana chilena reducida',redim) | En muchas ocasiones, las imágenes capturadas tienen dimensiones extensas, por lo que se les aplica un tamaño de 400x350 píxeles para reducir su dimensión. |


***Figura 3.*** Captura de imagen reducida a 400x350 sin perder calidad.

En la Tabla 3, al momento de ejecutar el comando mandamos una orden que se reduzca a escala de pixel establecido, así mismo, en la Figura 3 al instante aparece la imagen reducida dependiendo de que manzana hallamos colocado.

Recorte de región redimensionada, además, al observar que la imagen está reducida al máximo, a continuación, en la tabla 4 procederemos a recortar únicamente la parte de la manzana que deseamos identificar, de modo que no incluya demasiado borde de fondo (Manav, 2022).

**Tabla 4.**
*Recorte de imagen redimensionada*

| ***RECORTE*** | ***DESCRIPCION*** |
|---|---|
| recorte = redim[40:310,100:270]<br>cv2.imshow('imagen manzana chilena recorte',recorte) | Este código realiza un recorte de la imagen 'redim' con coordenadas específicas, seleccionando una región que corresponde a la parte de la manzana que se desea identificar. |

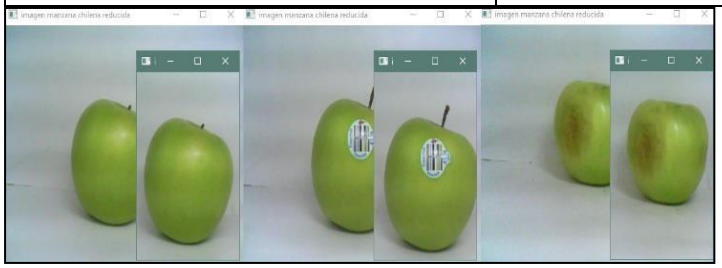

***Figura 4.*** Imagen recorte específico.

Podemos notar en la Figura 4, el recorte notorio solamente de la imagen que queremos identificar, en ello, solo hacemos un recorte específico del centro de la imagen.

Espacio de color HSV, este fragmento presentado en la tabla 5, es el comando que convierte la imagen recortada 'recorte' de formato RGB a formato HSV utilizando la función cv2.cvtColor() de OpenCV. Luego, muestra la imagen resultante en el espacio de color HSV en una ventana con el título 'Imagen de manzana chilena HSV' utilizando cv2.imshow(). El espacio de color HSV es útil para el análisis de color en imágenes (Akhtar Khan, 2022)

**Tabla 5.**
*Comando para generar a formato HSV*

| ***RGB a HSV*** | ***DESCRIPCION*** |
|---|---|
| imghsv=cv2.cvtColor(recorte,cv2.COLOR BGR2HSV)<br>cv2.imshow('Imagen de manzana chilena HSV', imghsv) | El comando convierte una imagen recortada (en formato RGB) de una manzana chilena a formato HSV. |

Al analizar la Tabla 5, se hace evidente que cada comando utilizado en Python está acompañado por un procedimiento específico. Esta relación entre comandos y procedimientos se aclara aún más en la figura 6, donde se muestran las paletas de colores correspondientes, la representación visual proporciona una comprensión más completa de cómo se aplican los comandos en Python y cómo se traducen en elementos visuales concretos.

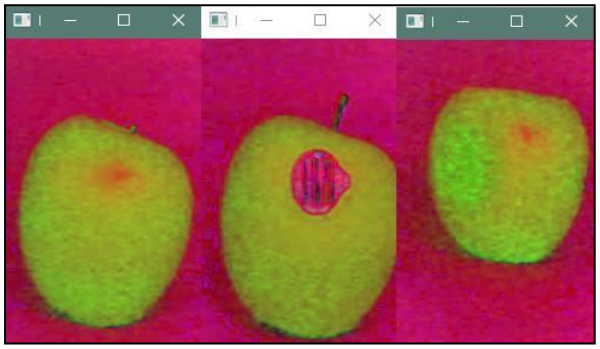

***Figura 6.*** Imagen de paleta de colores RBG a HSV.

Definición de rangos tono de color, En este procedimiento, vamos a definir el rango de color verde para la detección de la manzana en la imagen HSV, esto quiere decir que el programa Python detectara solamente el color verde rechazando los demás notos de colores (OMES, 2020).

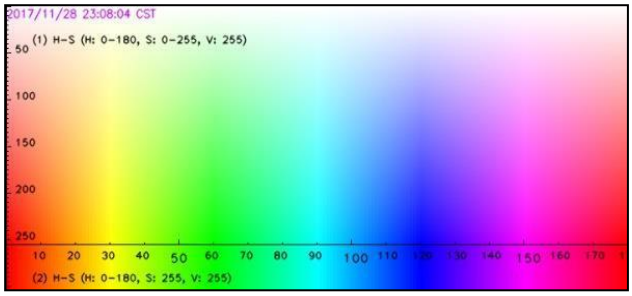


***Figura 7.*** Detección de colores en OpenCV - Python.

**Tabla 6.**
*Comando detecta tono de color y aplica umbral*

| ***TONOS DE COLOR*** | ***DESCRIPCION*** |
|---|---|
| verdebajo=np.array([20,30,30],np.uint8)<br>verdealto=np.array([60,255,255],np.uint8)<br>mascara1=cv2.inRange(imghsv,verdebajo,ve rdealto)<br>cv2.imshow('imagen manzana chilena imumbral',mascara1) | Este código en Python utiliza OpenCV para identificar y resaltar un rango de colores específico en una imagen en formato HSV, creando una máscara binaria y mostrando solo los píxeles que caen dentro de ese rango. |

Para identificar una manzana chilena de color verde, estableceremos un rango de valores en el canal de matiz de la imagen HSV, específicamente entre 20 y 60, basándonos en la figura 7 y la tabla 6. Utilizaremos este rango para crear una máscara de color que resalte la tonalidad verde de la manzana, como se muestra en la Figura 8. Posteriormente, aplicaremos un umbral a esta máscara para mejorar su definición.

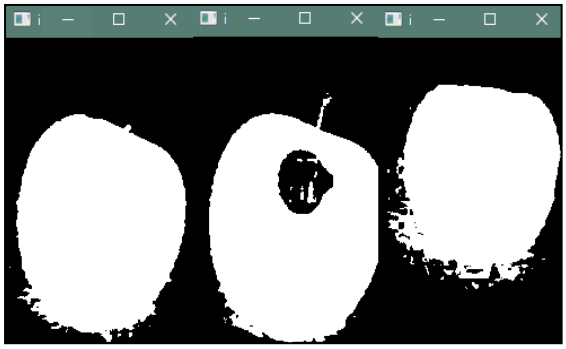

***Figura 8.*** Aplicación Umbra de imagen HSV.

Aplicación de erosión y dilatación, cuando se aplica mascara morphology el programa Python es capaz de generar erosión y dilatación de manera inmediata mejorar la calidad de una máscara binaria 'mascara1' (Toolify.ai,2024). En la tabla 7 se define un kernel de tamaño 3x3 con valores unitarios. Finalmente, la imagen resultante se muestra en una ventana con el título "Imagen manzana erosión y dilatación", tal como se presenta en la figura 9. Este proceso es útil para eliminar pequeños agujeros y conectar componentes en una imagen binaria, mejorando así la calidad y coherencia de la imagen procesada. La aplicación de estas técnicas de procesamiento de imágenes es crucial en diversas áreas, como la visión por computadora y el análisis de imágenes, donde es fundamental obtener resultados precisos y detallados.

**Tabla 7.**
*Comando aplica erosión y dilatación de imagen*

| ***EROSION Y DILATACION*** | ***DESCRIPCION*** |
|---|---|
| kernel=np.ones((3,3),np.uint8)<br>closing=cv2.morphologyEx(mascara1,cv2.M<br>(Programacion: ORPH_CLOSE,<br>kernel,iterations=3)(aPtiroongsr=a3m)<br>cv2.imshow('imagen manzana chilena erosion y dilatacion',closing) | Este código en Python mejora una máscara binaria llamada 'mascara1' utilizando operaciones morfológicas. Se realiza un cierre con un núcleo de 3x3 y tres iteraciones, y luego se muestra el resultado en una ventana. |

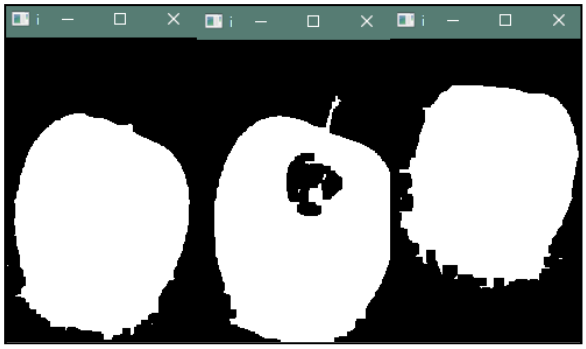

***Figura 9.*** Aplicación erosión y dilatación de imagen umbral.

Aplicación de suavizado y contornos, en la fase final de la transformación a escala de blanco y negro, se ejecuta un proceso de suavizado para difuminar la imagen, seguido de la detección y delineado de los contornos exteriores de la manzana. Se realiza un meticuloso recuento de los contornos identificados (Omes, 2019), cómo se detalla exhaustivamente en la tabla 8 para su referencia precisa. Como culminación, se presentan tanto la imagen suavizada como la imagen con los contornos marcados en la figura 9, ofreciendo así una representación visual completa y detallada del procedimiento.

**Tabla 8.**

*Comando aplica suavizado y marca contornos*

| ***SUAVIZADO Y CONTORNOS*** | ***DESCRIPCION*** |
|---|---|
| blur=cv2.blur(closing,(5,5))<br>contornos,jerarquia=cv2.findContours(blur,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)<br>cv2.drawContours(recorte, contornos,-1,(255,0,0),2)<br>numcont=len(contornos[0])<br>print('contornos',numcont) cv2.imshow('imagen manzana chilena suavizada',blur)<br>cv2.imshow('contornos manzana chilena',recorte) | Este código utiliza la biblioteca OpenCV en Python para realizar varias operaciones en una imagen.<br>Primero, aplica un filtro de desenfoque a una imagen llamada "closing" y luego encuentra los contornos en esa imagen suavizada. Los contornos encontrados se dibujan en la imagen original "recorte" y se cuenta cuántos contornos se encontraron.<br>Finalmente, muestra las imágenes procesadas en una ventana. |

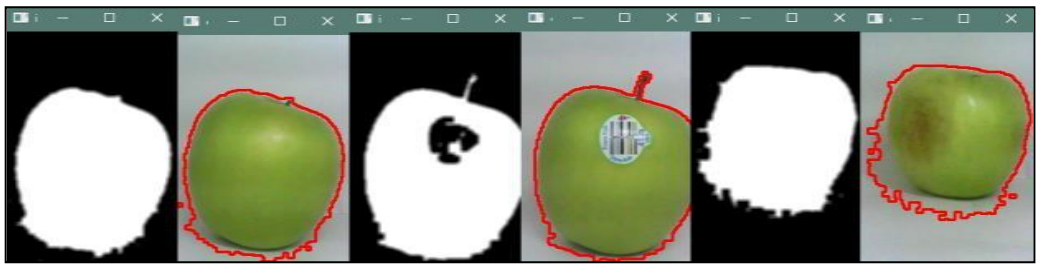

***Figura 9.*** Aplicación de suavizado y contornos

Calculación de momentos y áreas, dependiendo de los contornos encontrados, se calculan las posibles coordenadas en (X, Y), como se muestra en la tabla 9. El comando ‘‘print(cX)-print(cY)’’ imprime en la imagen, dibujando un círculo en el posible centroide del contorno. Esto se puede visualizar en la figura 10, donde se puede notar que en ciertas ocasiones puede ubicar el centro de la manzana, pero esto no significa que no cumpla su función de detección. Este método permite identificar y marcar el centroide de los contornos, facilitando la localización precisa de los objetos en la imagen. Además, proporciona una herramienta útil en el análisis de imágenes, ya que permite resaltar áreas de interés y evaluar la precisión de los algoritmos de detección implementados (Rodriguez, 2016).

**Tabla 9.**

*Comando calcula momentos y áreas*

| ***CENTROS, CONTORNOS Y ÁREAS*** | ***DESCRIPCION*** |
|---|---|

| | |
|---|---|
| cnt=contornos[0] M=cv2.moments(cnt) print(M)<br>cX=int(M["m10"] /M["m00"])<br>cY=int(M["m01"] /M["m00"])<br>print(cX);print(cY) cv2.circle(recorte,(cX,cY),2,(0,0,0),-1)<br>cv2.putText(recorte,"<br>x:"+str(cX)+",y:"+str(cY),(cX,cY),1,1,(0,0,0),2)<br>cv2.imshow("posible centroide",recorte) | Se encuentra el posible centroide de un contorno en una imagen. Primero, seleccionas el contorno y luego calculas sus momentos para obtener el centroide, que representa su centro geométrico. Finalmente, muestras el centroide en la imagen junto con sus coordenadas. |

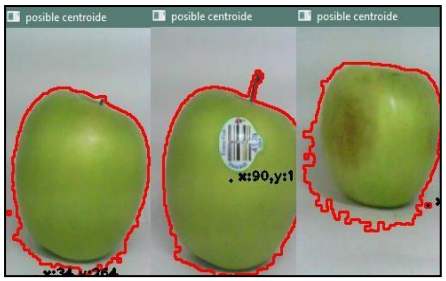


***Figura 10.*** Posibles centros en contornos de imagen

Detección de área y perímetros, en esta fase final se calcula el área y el perímetro de un contorno mostrado en la tabla 10, representado por cnt; Luego, evalúa si el perímetro es mayor que 599. Si es así, imprime "Manzana rechazada" y si es el perímetro por contornos en menor, se imprime “Manzana aceptada", recordemos que el perímetro supera un cierto umbral, la manzana se rechaza; de lo contrario, se acepta.

**Tabla 9.**
*Comando detección de áreas por perímetro >599*

| ***CENTROS, CONTORNOS Y ÁREAS*** | ***DESCRIPCION*** |
|---|---|
| area=cv2.contourArea(cnt) print(area)<br>perimetro=cv2.arcLength(cnt,True) print(perimetro)<br>if perimetro>599: print("Manzana rechazada")<br>cv2.putText (recorte, "Manzana rechazada", (10,260),<br>cv2.FONT_HERSHEY_SIMPLEX,0.6,(0,0,0))<br>cv2.imshow("Manzana rechazada",recorte)<br>else : print("Manzana aceptada")<br>cv2.putText (recorte, "Manzana aceptada", (10,260),<br>cv2.FONT_HERSHEY_SIMPLEX,0.6,(0,0,0))<br>cv2.imshow("Manzana aceptada",recorte) cv2.waitKey(0)<br>cv2.destroyAllWindows() | Este código en Python usando OpenCV analiza un contorno en una imagen, donde se calcula su área y perímetro, luego decide si el perímetro excede un umbral establecido. Si es mayor, se rechaza la manzana y se muestra un mensaje en la imagen. De lo contrario, se acepta la manzana y se muestra un mensaje. Al final, se presiona la tecla ‘‘Esc’’ para cerrar las ventanas de visualización. |

Resultado de manzanas en su estado, en nuestro ambiente controlado podemos colocar el tipo de manzana que deseamos identificar, si hemos puesto una manzana buena o mala, el mismo programa en Python mostrara todas las ventas que hemos realizado, esto lo pueden visualizar en la figura 11 y dentro de su historial IDLE Shell 3.11.8 en la figura 12 nos indicara los resultados obtenidos, donde mostrara los contornos encontrados, los momentos, centros con plano (X,Y) y por último si la manzana es aceptada o rechazada.

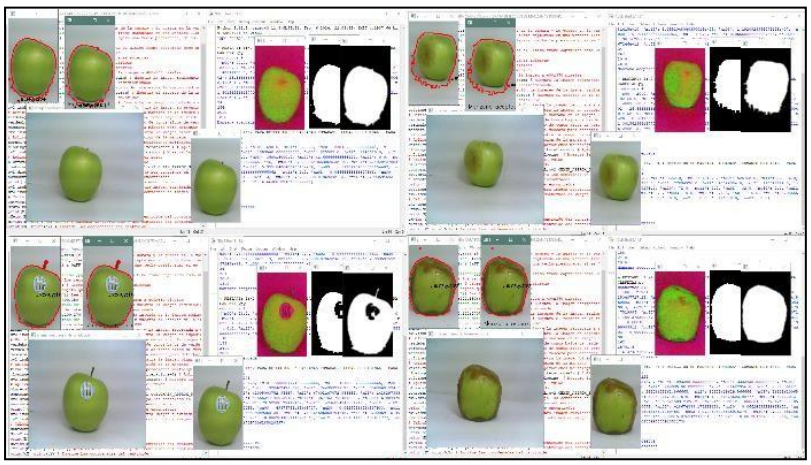

***Figura 11.*** Ventanas proyectadas por nuestra programación

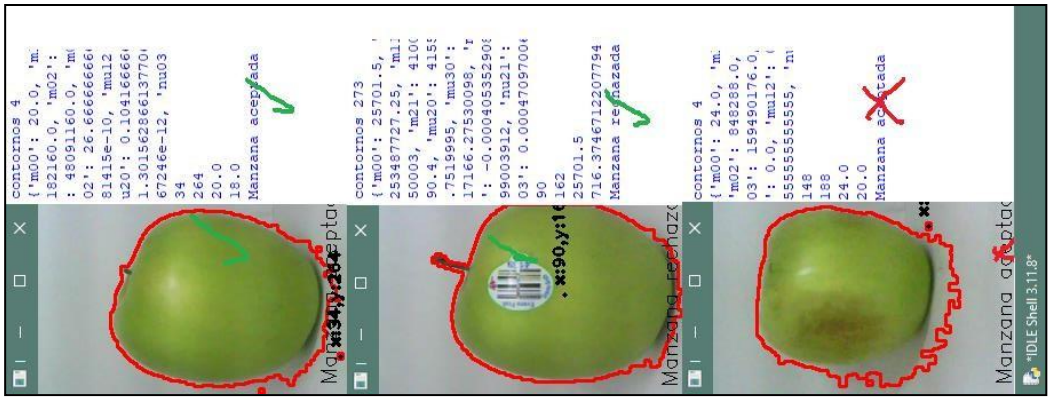

***Figura 12.*** Lista de resultados obtenidos por IDLE Shell Python

Recursos del proyecto, la Tabla 10 presenta una variedad de dispositivos y herramientas, como una laptop HP 15, una cámara web Jetion, una tira LED, un ambiente controlado y un programa Python con OpenCV, para investigación y análisis.

**Tabla 10.**
*Recursos usados para la realización del proyecto*

| ***MATERIALES*** | ***DESCRIPCION*** |
|---|---|
| Laptop HP 15 | • Procesador AMD Ryzen™ 3<br>• Windows 11 Home Single Language<br>• Unidad de 512 GB PCIe® NVMe™ M.2<br>• Una laptop potente y elegante para las tareas cotidianas. |
| Cámara web Jetion PJT-DCM141 | • Resolución de video 720p<br>• Interfaz: USB 2.0<br>• Compatibilidad con cualquier sistema operativo<br>• Cable longitud: 120 cm<br>• Reducción de ruido |
| Tira flexible LED | • Potencia: 50W / 10W por metro<br>• Resistencia calor: 3000k<br>• Lúmenes: 850Lm<br>• Frecuencia: 50-60Hz<br>• Tiempo de uso: 45,000Hrs<br>• Recubrimiento siliconado<br>• Medida: 5000x10mm |
| Ambiente controlado | • Adquisición de imágenes (manzanas chilenas)<br>• Recubierto interior de color blanco<br>• Orificios pequeños entrada de luces LED<br>• Medidas fijas para la posición de Cámara Web |
| Variedad de manzanas chilenas | • Manzanas en buen estado<br>• Manzanas en estado de descomposición<br>• Distintas etapas de madurez |
| Programa Python | • Versión 3.11.8 (2024)<br>• Tamaño del archivo: 10,7MB<br>• Biblioteca 24.0 (Contiene OpenCV)<br>• Compatible con cualquier sistema operativo<br>• Adquisidor de ventanas<br>• API: Corrección soporte de unidades de formato |

Diagrama de flujo, se presenta de forma gráfica el detallado proceso de uso del programa, siguiendo los pasos escritos, tal como se ilustra en la Figura 13.

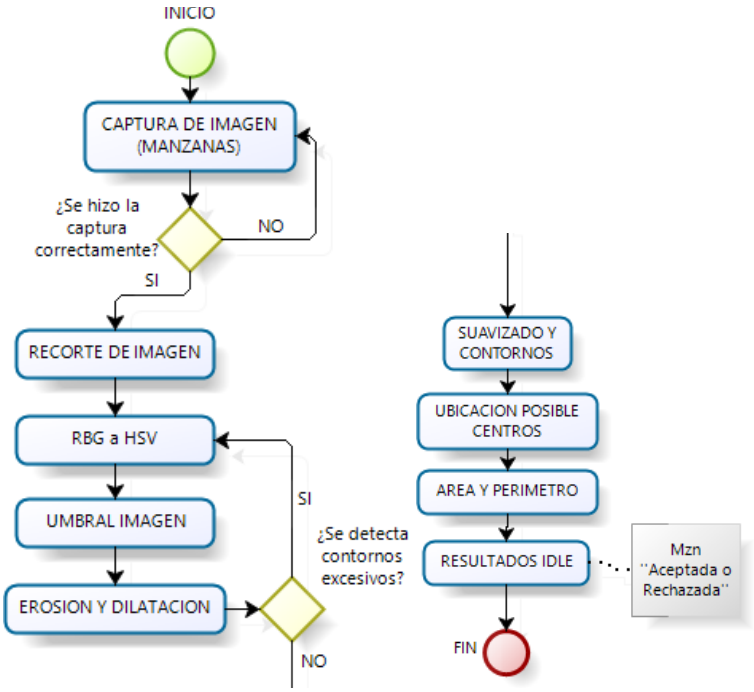


***Figura 13.*** Diagrama de flujo del procesamiento del proyecto

Esquema del proyecto final, la representación visual en la figura 14 proporciona un esquema detallado que ilustra el proceso completo, paso a paso, mediante el cual la programación permite identificar manzanas en buen estado y en mal estado, este proceso implica que la inteligencia artificial examina la imagen, utilizando algoritmos especializados para reconocer y analizar las características distintivas, tales como su color, forma y textura.

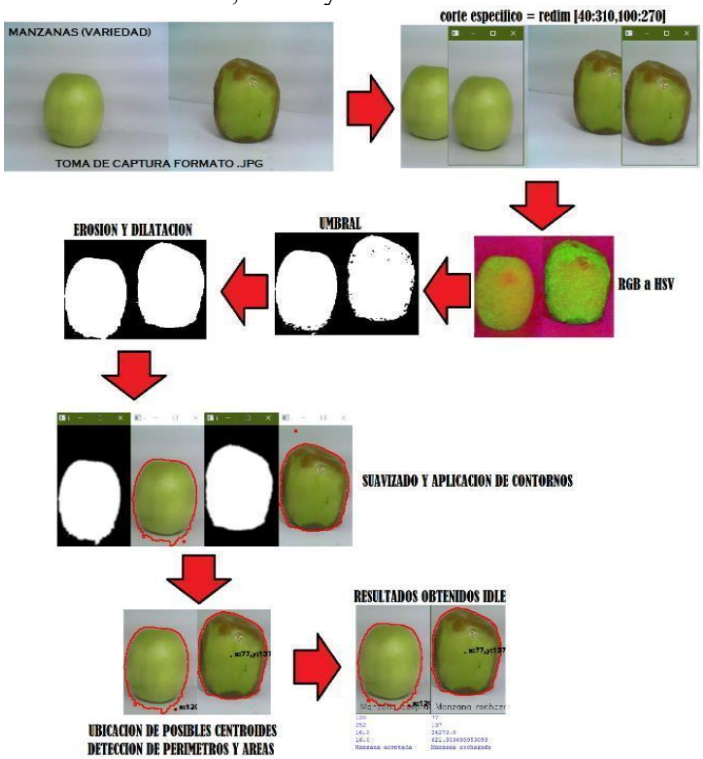


***Figura 14***. Esquema visual con imágenes reales

## Resultados

A continuación, se presentan los resultados de la programación realizada en Python, donde se tomaron de 20 a 57 fotos el cual evaluaremos la eficiencia y precisión de nuestro programa. Para ello haremos uso de predicciones del programa y predicciones visuales para saber qué tanto de porcentaje hemos obtenido, para ello, debemos sacar los resultados de las 57 tomas, así como se muestra figura 15


***Figura 15.*** Resultados de manzanas aceptadas o rechazadas

A partir de los resultados presentados en la figura 15, podemos llevar a cabo nuestra investigación, utilizando la precisión para evaluar qué tan exacto es nuestro programa en comparación con el ojo humano. Aplicaremos ecuaciones matemáticas para calcular estos resultados, los cuales se reflejan en la figura 16. Las precisiones obtenidas se detallan en la tabla 11, lo que nos permitirá determinar la precisión de nuestro programa en un entorno controlado.

### *Predicciones programa*

TMP = Total manzanas positivas (Programación) TMD = Total manzanas con deterioro (Programación)

### *Predicciones visual*

TMA = Total de manzanas aceptadas (Visual) TMR = Total de manzanas rechazadas (Visual)

**Tabla 11.**
*Tablas de Precisión*

| *MUESTRA* | *T.M.P* | *T.M.D* | *T.M.A* | *T.M.R* | *PRECISION* |
|---|---|---|---|---|---|
| 20 | 14 | 5 | 18 | 3 | 90% |
| 57 | 50 | 7 | 54 | 6 | **95%** |

Incluso, aunque el programa cumpla su función queremos saber que tan eficiente es para detectar manzanas en buen estado, donde lo pondremos a prueba con manzanas en descomposición para saber que tan eficiente se muestra, esto lo podemos ver en los resultados de la tabla 12.

**Tabla 12.**
*Tablas de Eficiencia*

| *MUESTRA* | *ACEPTADA* | *RECHAZADA* | *ERROR* |
|---|---|---|---|
| 1 | x | | NO |
| 2 | x | | NO |
| 3 | x | | NO |
| 4 | | i | NO |
| 5 | x | | NO |
| 6 | | i | NO |
| 7 | x | | NO |
| 8 | x | | NO |
| 9 | x | | NO |
| 10 | | i | NO |
| 11 | x | | NO |

| | | | |
|---|---|---|---|
| 12 | x | | NO |
| 13 | | i | NO |
| 14 | x | | NO |
| 15 | | | SI |
| 16 | | i | NO |
| 17 | x | | NO |
| 18 | x | | NO |
| 19 | x | | NO |
| 20 | x | | NO |
| 21 | x | | NO |
| 22 | x | | NO |
| 23 | x | | NO |
| 24 | x | | NO |
| 25 | x | | NO |
| 26 | x | | NO |
| 27 | x | | NO |
| 28 | x | | NO |
| 29 | x | | NO |
| 30 | x | | NO |
| 31 | x | | NO |
| 32 | x | | NO |
| 33 | x | | NO |
| 34 | x | | NO |
| 35 | x | | NO |
| 36 | x | | NO |
| 37 | x | | NO |
| 38 | x | | NO |
| 39 | x | | NO |
| 40 | x | | NO |
| 41 | x | | NO |
| 42 | x | | NO |
| 43 | x | | NO |
| 44 | x | | NO |
| 45 | x | | NO |
| 46 | | | SI |
| 47 | x | | NO |
| 48 | x | | NO |
| 49 | x | | NO |
| 50 | x | | NO |
| 51 | x | | NO |
| 52 | x | | NO |
| 53 | x | | NO |
| 54 | x | | NO |
| 55 | x | | NO |
| 56 | x | | NO |
| 57 | x | | NO |

| **CONTEO** | |
|---|---|
| ACEPTADAS | 50 |
| RECHAZADAS | 5 |
| ***TOTAL (Acept, Rechz)*** | **55** |
| ERROR | 2 |
| ***TOTAL*** | ***57*** |
| Eficiencia del proyecto | 0.9298 |

| Porcentaje obtenido | **93%** |
|---|---|

## Discusión

Mediante la programación se pudo comprobar la eficiencia y la precisión del análisis de manzanas chilenas. Se recalca que al aplicar diferentes muestras el proyecto genero una eficiencia del 93% y una precisión del 95% al realizar adquisición de imágenes, así también a comparación a Ohali (Al Ohali, 2010), obtuvo una precisión del 80% en clasificación de frutas por computador mientras que nuestros resultados son mayores. De manera similar, para José Bravo en su proyecto de detección del daño causado por el gusano cogollero en plantas de maíz logro un 82.6% de eficiencia (Bravo, 2020), en comparación nuestros resultados arrojan valores superiores. En otra investigación, Huillca desarrolló un sistema de clasificación para aguaymanto y fresas (Huillca, 2022), el cual sus resultados lograron una precisión del 91.33% para aguaymanto y del 90.5% para fresas en comparación a nuestro proyecto de 95% notamos superioridad en precisión. Además, en la investigación por parte de Burgos Zavaleta, Ulloa Baquedano, Aguilar Acevedo, Robles Malqui y Julián Suarez (2022); crearon un sistema de visión artificial para detectar abolladuras en latas solo realizaron cinco series de pruebas con diez latas logrando solo una eficiencia del 90.4%, con respecto a nuestro proyecto, en eficiencia lo superamos con 93% (Burgos Zavaleta et al., 2022).

## Conclusiones

Hemos logrado un importante avance tecnológico al detectar manzanas chilenas en buen estado. Con el esfuerzo conjunto de nuestro grupo y la ayuda del docente, este avance no solo mejora la eficiencia en el curso de automatización, sino que también contribuye significativamente a la salud de muchos peruanos.

Los algoritmos empleados son clasificados por color, tamaño y perímetro para detectar el estado deseado, dando también lugar mejorar muchas empresas interesadas en el lenguaje de programación.

Los resultados obtenidos fueron satisfactorios, como se muestran en las tablas de un total de 57 tomas, resulto mostrar una precisión de 95% con una eficiencia del 93%.

En conclusión, el uso de Python con OpenCV requiere un enfoque que tome más tiempo para obtener resultados óptimos. Creemos que, con mejores herramientas de adquisición de imágenes, los resultados serían de alta calidad.

## Referencias

#SomosNoticia. (2021, 13 de agosto). SENASA y productores del valle de Chicama intensifican acciones para producir guabas, uvas, naranjas y manzanas libres de la plaga mosca de la fruta. Servicio Nacional de Sanidad Agraria. https://www.senasa.gob.pe/senasacontigo/senasa-y-productores-del-valle-de-chicama-intensifican-acciones-para-producir-guabas-uvas-naranjas-y-manzanas-libres-de-la-plaga-mosca-de-la-fruta/

Akhtar Khan, S. (2022, 27 de septiembre). How to find the HSV values of a color using OpenCV Python? https://www.tutorialspoint.com/how-to-find-the-hsv-values-of-a-color-using-opencv-python

Al Ohali, Y. (2010). Computer vision based date fruit grading system: Design and implementation. King Saud University.

Bravo Reyna, J. L. (2020). Detección automática del daño causado por el gusano cogollero en la planta de maíz [Tesis de licenciatura, Tecnológico Nacional de México].

Burgos Zavaleta, P. A., Ulloa Baquedano, J. F., Aguilar Acevedo, J., Robles Malqui, A. R., & Julián Suarez, C. (2022). Desarrollo de un sistema de visión artificial para detección de abolladuras en latas de atún. Universidad Privada del Norte.

ComexPerú. (2023, 15 de diciembre). Aún en rojo: Las exportaciones peruanas cayeron un 0.1% a octubre de 2023. https://www.comexperu.org.pe/articulo/aun-en-rojo-las-exportaciones-peruanas-cayeron-un-01-a-octubre-de-2023

Cueva Caro, J. E. (2022). Redes neuronales convolucionales para determinar la acidez y grados Brix de Pitahaya amarilla (Hylocereus megalanthus) en la región Amazonas [Tesis de licenciatura, Universidad Nacional Toribio Rodríguez de Mendoza de Amazonas].

DataScientest. (2019, 18 de enero). NumPy: La biblioteca de Python más utilizada en Data Science. https://datascientest.com/es/numpy-la-biblioteca-python

Huillca Tumba, J. A. (2022). Diseño de un sistema de clasificación por categorías de aguaymanto y fresas usando redes neuronales convolucionales para el control de calidad en el proceso de deshidratado en la región de Huánuco [Tesis de licenciatura, Universidad Nacional Hermilio Valdizán].

INEI. (2023, 16 de mayo). Informe económico y comercial Perú. ICEX. https://www.icex.es/content/dam/es/icex/oficinas/065/documentos/2023/05/anexos/iec-peru-2023.pdf

IngiVision. (2021, 23 de marzo). Todo lo que debes saber sobre inteligencia artificial y su aplicación en la industria hortofrutícola. https://www.ingivision.com/2021/03/23/inteligencia-artificial/

La República. (2019, 10 de octubre). Tacna: Decomisan 60 kilos de fruta malograda en mercado. https://larepublica.pe/sociedad/2019/10/10/tacna-decomisan-60-kilos-de-fruta-malograda-en-mercado-fiscalia-lrsd

León León, R. A., & Garcia Saire, J. M. (2018). Sistema de visión artificial para la inspección de fechas en productos industriales alimenticios. Universidad César Vallejo.

Manav, N. (2022, 22 de enero). Recortar imagen usando OpenCV en Python. https://www.delftstack.com/es/howto/python/crop-images-using-opencv-in-python/

MAPA. (2020, 15 de julio). Perú está entre las potencias exportadoras de fruta en el mundo (p. 1). Instituto Nacional de Estadística e Informática

Martin Monroy, M. (2006). Manual de la iluminación. Editorial ANAYA Multimedia.

Martínez, J. (2022, 7 de enero). Cómo cambiar el tamaño de una imagen en OpenCV. https://www.datasmarts.net/como-cambiar-el-tamano-de-una-imagen-en-opencv/

Omes. (2019, 17 de octubre). Contornos y cómo dibujarlos en OpenCV y Python. https://omes-va.com/contornos/

OMES. (2020, 3 de enero). Contando por colores en Python-OpenCV. https://omes-va.com/contando-por-colores-en-python-opencv/

PlazaVea. (2024, 1 de enero). Manzana verde importada. https://www.plazavea.com.pe/manzana-verde-importada/p

Rivera Vaca, C. D. (2023). Sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de visión artificial [Tesis de licenciatura, Universidad Técnica del Norte].

Rodriguez Ojeda, L. (2016). Python programación. Ediciones Universitarias.

Sepúlveda, L. A., & Araya, Y. J. (2017). Efecto del clima en la calidad de las manzanas. Boletín del Centro de Pomáceas, 17(1), 4–6.

Serrano Fuente, M., Lizardo Zelaya, N. A., & Ordoñez Avila, J. L. (2020). Coffee fruit recognition using artificial vision and neural networks. Universidad Nacional Autónoma de Honduras.

Toolify.ai. (2024, 10 de febrero). Aprende sobre erosión y dilatación de imágenes en niveles de gris. https://www.toolify.ai/es/ai-news-es/aprende-sobre-erosin-y-dilatacin-de-imgenes-en-niveles-de-gris-1082951