Desarrollo de sistema de clasificación de color y apariencia de chontaduro para exportación en la finca el Cedro de Villa Garzón (Putumayo) utilizando procesamiento de imágenes

Jaiderne Marín Rojas

20441718375

Universidad Antonio Nariño

Programa Ingeniería Electrónica

Facultad de Ingeniería Mecánica, Electrónica y Biomédica

Ibagué, Colombia

2023

Desarrollo de sistema de clasificación de color y apariencia de chontaduro para exportación en la finca el Cedro de Villa Garzón (Putumayo) utilizando procesamiento de imágenes

Jaiderne Marín Rojas

Proyecto de grado presentado como requisito parcial para optar al título de:
Ingeniero Electrónico

Director (a):
Ing. José Fernández

Línea de Investigación:
Nombrar la línea de investigación en la que se enmarca el trabajo de grado.

Universidad Antonio Nariño

Programa Ingeniería Electrónica

Facultad de Ingeniería Mecánica, Electrónica y Biomédica

Ibagué, Colombia

2023

NOTA DE ACEPTACIÓN

El trabajo de grado titulado **Desarrollo de sistema de clasificación de color y apariencia de chontaduro para exportación en la finca el Cedro de Villa Garzón (Putumayo) utilizando procesamiento de imágenes**

, Cumple con los requisitos para optar

Al título de Ingeniero Electrónico.

_____

Firma del Tutor

_____

Firma Jurado

_____

Firma Jurado

Ibagué, Octubre de 2023.

Contenido

Pág.

Lista de Figuras

Pág.

Lista de tablas

Pág.

(Dedicatoria)

A Dios por darme perseverancia y sabiduría para lograr esta meta tan importante en mi formación profesional.

A mis padres, Jaiderne y Maria Del Pilar, por el apoyo incondicional que me brindaron durante todo este proceso universitario, por ser quienes están ahí para darme ánimos de lograr cada una de mis metas.

Jaiderne Marín Rojas

Agradecimientos

Mi más sincero agradecimiento a mi director Ingeniero Ing. José Fernández por su confianza y acompañamiento desde el inicio de este proyecto, quien siempre estuvo al pendiente de mi proceso y me brindó todo su apoyo y conocimiento para hacer esto posible, a la ingeniera Andrea Marín por su ayuda y respaldo al inicio de mi proyecto, ya que estuvo al pendiente y confió en mis habilidades, agradezco a la Universidad y a todo el plantel docente por haberme formado profesionalmente, a todos, muchas gracias

Jaiderne Marín Rojas

Resumen

El procesamiento de imágenes para uso en proceso de clasificación de frutas es un método efectivo para mitigar devoluciones en su proceso de exportación, esta solución no solo aporta en la clasificación, si no en la simplificación de errores , costos y daño del producto; el presente proyecto tiene como fin desarrollar el sistema de clasificación de color y apariencia de chontaduro para exportación, en la finca el cedro de Villa Garzón (Putumayo) mediante procesamiento de imágenes. La metodología empleada incluye el proceso de adquisición de imágenes mediante una cámara Web C505 HD, procesada mediante lenguaje Python y segmentada mediante códigos en librerías OpenCV a partir de parámetros de clasificación (verde, rojo, naranja, negro) permiten identificar el nivel de madurez de la fruta. Los criterios establecidos comprenden rangos entre 0 y 30% rechazada y de 70 a 90 % fruta aceptada o clasificada

Palabras clave: Chontaduro, procesamiento digital de imágenes, grado de madurez

Abstract

The Image processing for use in fruit classification process is an effective method to mitigate returns in the export process, this solution not only contributes in the classification, but also in the simplification of errors, costs and product damage; this project aims to develop the classification system of color and appearance of chontaduro for export, in the farm the cedar of Villa Garzón (Putumayo) through image processing. The methodology used includes the process of image acquisition using a C505 HD Web camera, processed using Python language and segmented using codes in openCV libraries based on classification parameters (green, red, orange, black) to identify the level of maturity of the fruit. The established criteria include ranges from 0 to 30% rejected and from 70 to 90% fruit accepted or classified.

Key words: Chontaduro, digital image processing, degree of maturity.

Introducción

El chontaduro pertenece a la familia de las aceráceas y es una fruta de tipo cónica, ovoide o elíptica; crece en el cogollo en forma de drupas o racimos, en palmas de una altura máxima de 20 metros (FINAGRO, 2023); el proceso de recolección se realiza de forma manual con el uso de maroteo, cuyo principio de funcionamiento consiste en escalar la palma apoyado por lazos que aseguran al operario en su ascenso hasta el racimo; aquí, el operario realiza la recolección (Chamorro, 2018).

Este cultivo es uno de los motores de la agricultura en la región de Villa Garzón (Putumayo) y existe una oportunidad, de poder mejorar las prácticas de clasificación del producto mediante nuevas tecnologías. Este trabajo centra su investigación en generar una alternativa tecnológica como punto de partida para el control de calidad de la finca el Cedro utilizando procesamiento de imágenes.

El 70% de producción de chontaduro tiene como destino el mercado internacional, al cual se destina el fruto clasificado por color rojo, naranja, amarillo y buena apariencia que refiere a un producto no averiado por cortes.

Los mercados a nivel local y mundial son cada vez más exigentes en cuanto a la calidad de productos que demandan los consumidores; lo que conlleva a que se establezcan estándares para su comercialización para satisfacer diferentes mercados en función a parámetros como peso, tamaño, color, estado de madurez entre otros. Debido a esto las actividades de clasificación y selección en la postcosecha tienen un papel fundamental para cumplir con los requerimientos exigidos por los compradores.

El trabajo presentado a continuación tiene como objetivo desarrollar un sistema de clasificación de color y apariencia de chontaduro para exportación, en la finca el cedro de Villa Garzón (Putumayo) mediante procesamiento de imágenes; el documento se compone de 5 capítulos los cuales parte de la descripción del problema, seguido del marco teórico, el diseño metodológico, un capítulo 4 que aborda el desarrollo del hardware y software y por ultimo los resultados.

1. Capítulo 1

1.1.Planteamiento del problema

El proceso de recolección de chontaduro en la finca el Cedro de Villa Garzón se realiza de forma manual, esto debido a que la planta que produce el chontaduro es una palma que alcanza alturas que van desde los 12 hasta los 20 metros de altura (CORPOICA, 2021), razón por la cual el proceso es manual e implica que los trabajadores escalen la palma para realizar el corte del chontaduro el cual es descolgado por gravedad, momento en el que puede tener contacto con la superficie y dañar su apariencia. Se calcula una relación en la que por cada 20 frutas se pueden cizallar 5 chontaduros en esta finca.

A nivel nacional la producción de chontaduro es de alrededor de 250 Tn/mes (Torres, 2022), de las cuales se devuelven alrededor del 12 % por concepto de producto averiado y color. Esto deja en evidencia la existencia de errores de tipo humano en el proceso que impactan a nivel de costo al productor y afecta la confiabilidad y calidad de la fruta en el mercado.

De acuerdo con lo anterior se plantea la siguiente pregunta de investigación

¿Es posible mejorar el proceso de clasificación de chontaduro en función de las exigencias de color y apariencia para mercados destinos y de esta manera reducir la tasa de devolución de este utilizando procesamiento de imágenes?

1.2.Antecedentes

A continuación, se presentan algunos artículos científicos que caracterizan los avances tecnológicos en cuanto a sistemas de clasificación por color y apariencia.

1.2.1. Clasificación por Color

Para Patil et al (2021), el uso de técnicas de clasificación para la pitaya o fruta del dragón, a partir del uso de algoritmos de aprendizaje automático (CNN, ANN y SVM) permite ampliar con mayor exactitud el uso de otros parámetros como tamaño, peso y color para la selección de esta; para el caso de la red neuronal artificial, esta consta de tres capas, una de entrada, una capa oculta y una capa de salida. La función de sesgo y los pesos de todos estos parámetros están conectados a las neuronas, las cuales se encargan de procesar los datos para el proceso de clasificación (Patil, 2021).

Menon et al (2021), plantea que el método de procesamiento de imágenes y aprendizaje profundo se convierte en una opción de automatización con un mínimo margen de error en procesos de clasificación de frutas, ya que emplea para su lógica estándares o parámetros técnicos precisos ajustados a un mercado en particular, lo que le proporciona al agricultor confiabilidad en la fruta comercializada (Menon, 2021).

Rodríguez & Salazar (2020), desarrollaron un prototipo de máquina para clasificación de piña en función del color y el tamaño por medio de visión artificial, la cual se basó en un sistema de transporte por rodillos que posiciona la fruta, para escaneo mediante cámara Raspberry Pi dentro de un sistema de iluminación controlado, el sistema de clasificación mediante el accionamiento de cilindros neumáticos y la salida de la fruta por bandejas. El algoritmo se desarrolló en Python mediante la utilización de librerías de

OpenCV. La clasificación tanto por color y forma se realizó mediante comparación de un valor denominado umbral. Se implementó una interfaz gráfica de usuario mediante una pantalla táctil y el control del prototipo se realizó mediante la Raspberry Pi. Para los resultados se realizó una comparación de la clasificación manual y la del prototipo (Rodríguez, 2020).

Ramírez et al (2018) utilizaron el procesamiento digital de imágenes para segmentar y analizar las características de color, en espacios RGB, de imágenes de la Ciruela Africana (Prunus africana); el método empleado incluye el uso de la propiedad de discontinuidad entre píxeles vecinos, detección de bordes y círculos. Además de morfología matemática. Para el desarrollo de este trabajo se utilizó el software Matlab (Ramírez, 2017) .

Pourdarbani et al (2015) desarrolló un sistema de clasificación de frutas de dátil en línea basado en visión artificial. Dicho sistema se compone de una banda transportadora, una unidad de iluminación y captura y una unidad de clasificación; los parámetros de clasificación (color, peso, tamaño, dimensiones) se determinaron a partir de muestras y el algoritmo de detección se diseñó en consecuencia; cuando la fruta se colocaba en el centro del campo de visión de la cámara, esta se escaneaba de forma instantánea para su procesamiento; cuando la fruta pasa por el sensor, ubicado al final de la cinta transportadora, se envió una señal al circuito de interfaz y se accionó un actuador apropiado, impulsado por un motor paso a paso, que condujo al Date hacia un puerto apropiado (Pourdarbani, 2018).

1.2.2. Clasificación por tamaño

Sin embargo Reyes (2021), desarrolló una célula de carga combinada con una plataforma Arduino para automatizar el proceso manual de selección de clasificar frutas de

lima-limón; este desarrollo usó un motor que controla el proceso de clasificación y la velocidad de transporte de la fruta sobre la banda, un sensor de celda de carga para pesar los limones clasificados y una placa Arduino Nano, la cual envía los datos al sistema de la aplicación de escritorio, lo que permite visibilizar los limones clasificados y el tamaño de los mismos junto con su peso (Reyes, 2021).

Muñoz & Casallas (2020), desarrollaron una máquina para la clasificación de tomate Chonto por tamaño y color a partir de un proceso de transporte, el cual lleva la fruta a una posición definida, en donde el sistema de visión toma una fotografía del producto para realizar el tratamiento de la imagen, para así determinar las características de este, luego por medio de varios actuadores cada producto es lanzado a una bandeja de despacho designada en algoritmo según parámetros técnicos establecidos en la NTC, lo que representa disminución de costos, tiempos y calidad en el producto para el agricultor (Muñoz, 2021).

1.3.Objetivos

1.3.1. General

Desarrollar el sistema de clasificación de color y apariencia de chontaduro para exportación, en la finca el cedro de Villa Garzón (Putumayo) mediante procesamiento de imágenes.

1.3.2. Específicos

- Establecer las variables de clasificación por color y aspecto exterior (apariencia) del chontaduro.
- Dimensionar los requerimientos del sistema (clasificación color y apariencia) de clasificación de Chontaduro en la finca el Cedro de Villa Garzón (Putumayo).
- Diseñar el sistema de procesamiento para la clasificación de chontaduro.
- Diseñar el sistema de comunicación.
- Evaluar la efectividad del sistema de clasificación por color y apariencia en función de errores de clasificación.

1.4.Justificación

La automatización de procesos permite mejorar la efectividad de las actividades, así como el control de calidad, permitiendo optimizar recursos humanos, financieros y técnicos a largo plazo, a su vez otras variables de control-proceso fundamentales para los productores.

A esto se suma que la automatización le permite al productor, estandarizar las actividades, permitiéndole optimizar el tiempo de estas, lo cual incide en la asignación de nuevas tareas para el personal disponible, así como la optimización de costos de producción.

La presente investigación propone una opción viable de mejorar la eficiencia de clasificación del chontaduro en la finca el Cedro de Villa Garzón (Putumayo) a partir del diseño de un sistema por captura de imágenes que permitirá categorizar el producto en función del color y la apariencia con miras a bajar las tasas de devolución de este con destino al mercado internacional.

La pertinencia del proyecto parte del hecho de poder mejorar proceso en donde los errores humanos impactan a los productores; a su vez este tipo de implementaciones puede beneficiar a otros cultivadores en el territorio nacional, lo que permite fortalecer a nivel nacional la cadena productiva y visibilizando las bondades del uso de tecnologías en el campo colombiano.

2. Capítulo 2: Marco teórico

2.1.Chontaduro

El chontaduro es propio de las áreas tropicales, adaptable en zonas de alta precipitación y temperatura, se cultiva en altitudes promedios que oscilan entre los 100 a 800 m.s.n.m, especialmente en topografías mixtas que incluyen terrenos quebrados y planos a excepción de los húmedos o muy compactos, ya que inducen la madurez de la fruta o retrasan su desarrollo o en el caso extremo la planta no crece lo necesariamente. El chontaduro se caracteriza por sus nutrientes, contiene 37,6% de carbohidratos; 52,5% de agua; 4,6% de grasa; 3,3% de proteínas; 1,4% de fibra; 23 mg de calcio; 47 mg de fósforo; 0,7% de hierro y aportar 185 calorías/g, razón por la cual es una fruta muy deseada y con alto poder nutricional (Daza, 2015).

En cuanto a las variedades de chontaduro en el área de estudio se tiene 2 tipos como se indica en la tabla 1-1.

Tabla 1-1. Variedades de Chontaduro en la finca

Variedad	Características
1	Frutos con pigmentación naranja y forma cónica
2	Frutos con pigmentación roja y forma cónica achatada

Fuente: Propietario finca

Estas variedades son comunes en la zona de estudio, como se observa en la figura 1-1 su forma puede variar en función de los parámetros de la tabla 1-1, normalmente esta fruta se da en forma de drupas o racimos los cuales se cosechan cuando la fruta esta de color rojo o naranja, en algunas ocasiones su color no es homogéneo, razón por la cual cuando más del 70 % de la fruta alcanza esa apariencia roja o amarilla, los racimos son descolgados para su posterior clasificación y comercialización.

Figura 1-1. Frutas de chontaduro

Fuente: Propietario finca

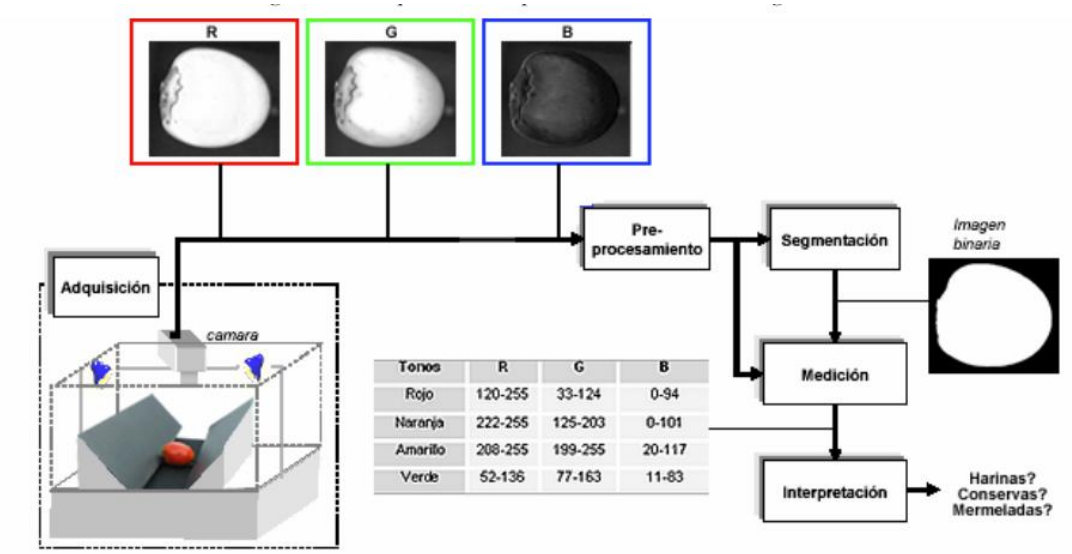
2.2.Procesamiento digital de imágenes

Uno de los aspectos más incidentes del uso de tecnologías de reconocimiento, se asocia al procesamiento de imágenes digitales (PDI), ya que estas constituyen hoy día un lenguaje de comunicación que simplifica de manera precisa una tarea determinada en cualquier campo de uso; esta tarea parte de una de convolución de la imagen, a partir de una lectura de escalas de grises la cual permite definir de forma concreta el concepto de la imagen (UNC, 2023).

En lo que respecta a la escala de grises esta emplea para su definición tres fases las cuales, con el resaltamiento de la imagen, la minimización del ruido y la identificación de bordes; en la primera fase se busca que dicha imagen tenga una mejor definición a partir de aspectos físicos y cognitivos que tenga el usuario, aquí es fundamental en número de pixeles que contiene la misma; la eliminación del ruido consiste en no considerar aquellos pixeles fuera del rango de la imagen; la tercera fase considera la detección de píxeles que generan modificaciones bruscas en la intensidad de la imagen (Mejía, 2014).

Las etapas del procesamiento imágenes parten del siguiente esquema general presentado en la figura 1-1

Figura 1-2. Etapas de procesamiento de imágenes



Fuente: Autor

2.2.1. Adquisición de imágenes

Al digitalizar la imagen bidimensional se pueden identificador sus componentes pixelados, los cuales representan áreas específicas de la imagen(Serrano et al., 2018: p.21).

Existen dos formas para el almacenamiento de imágenes a color, estas son las representaciones en formato RGB, aquí el tamaño del pixel es de 24 bits que contiene la cantidad de rojo (R), verde (G), y componentes azules (B), y las representaciones indexadas en las que matrices 2D contienen índices a una paleta de colores (Marques, 2011, p.25).

2.2.2. Procesamiento

El procesamiento emplea operaciones y transformaciones dadas a las imágenes de tipo digital, su fin es el de mejorar la resolución de la imagen para que se destaque en las

fases que comprenden la detección o identificación de los parámetros indicados o definidos por el usuario (Serrano T. , 2018)

Los procesos por lo general se llevan a cabo teniendo en cuenta los valores de píxeles en el espacio de color en el que están definido, en caso de que sean más colores se emplean métodos denominados espacios de color o modelos de color (García, 2015).

Así mismo el modelo RGB establece que la imagen se compone de tres planos o canales de imagen independientes: rojo, verde y azul (un cuarto canal para la transparencia, denominado canal alfa). En el modelo de escala de grises, el valor de cada píxel tiene un único valor sobre parámetros como información de intensidad, formando una imagen exclusivamente a partir de diferentes tonos de gris.

El modelo de color HSV se representan el tono H el cual da una medida de la composición espectral de un color, la saturación S que proporciona la luz pura de la longitud de onda dominante, determinan la cantidad de blanco que contiene un color, y V que da el brillo (García, 2015).

2.2.3. Segmentación

La segmentación es un proceso sistemático el cual se basa en dividir una imagen digital en áreas iguales en función a características como brillo o color. El resultado de dicha segmentación corresponde a una imagen en la que cada píxel tiene asignada una etiqueta distintiva del objeto al que pertenece (Serrano m. , 2018)

2.2.4. Reconocimiento o clasificación

En esta etapa se pretende distinguir los objetos segmentados, al ser esta la etapa final se cuenta con los análisis de ciertas características que se establecieron previamente para diferenciarlos. Determinadas las características discriminantes, se procede a la obtención del

patrón del objeto. Después se determina el grado de pertenencia del patrón a cada una de las clases, asignando el objeto a las clases con las que el grado de semejanza sea mayor (Serrano et al., 2018: p.138).

Lenguaje de programación Python

Python es un lenguaje programación dirigido a objetos, con una semántica dinámica integrada base para la construcción de páginas web y otras aplicaciones. Dentro de su uso básico este lenguaje se denomina «cpython«, razón por la cual emplea un código denominado byte, el cual puede ser interpretado por la CPU; de esta manera se requiere de traductor llamado Máquina Virtual Python (PVM) que ejecuta los códigos de bytes.

Dentro de las tareas de este interpretador se tienen:

- Paso 1: Lectura de código o instrucción Python; esta es verificada mediante un proceso de formateo o validación de sintaxis lineal. Si encuentra algún error, detiene inmediatamente la traducción y muestra un mensaje de error.
- Paso 2: Cuando no existe error, dicho código se conoce como «código Byte». Lo anterior permite un desarrollo exitoso de la escritura python.
- Paso 3: El código del byte se envía a la Máquina Virtual Python, donde de nuevo se ejecuta el código del byte en PVM. Si se produce un error durante esta ejecución, ésta se detiene con un mensaje de error.

Las librerías en Python:

Numpy: Empleada en procesos operacionales como algebra lineal, matrices y formas lógicas; emplea un lenguaje c++ y c, y en la creación de rangos para el procesamiento digital de imágenes.

Pandas: Emplea datos estructurales y de series temporales, matrices y tablas, se utiliza para sacar las matrices en procesamiento digital de imágenes.

2.3.Librería OpenCV

La librería OpenCV es una estructura única para visión por computador en tiempo real; permite la interacción hombre-máquina (HCI4); segmentaciones y reconocimientos de objetos, así como gestos, seguimiento del movimiento, estructura del movimiento (SFM5) y robots móviles (Arévalo, 2018).

OpenCV (Open Source Computer Vision) es una librería multiplataforma originalmente desarrollada como proyecto por Intel para apoyar a los primeros compiladores Intel C++ y Microsoft Visual C++ en x86.

Se usa con mucha frecuencia para procesar imágenes y proyectos de visión artificial, así como para controlar procesos operativos, sistema de seguridad mediante detección movimiento, identificación objetos entre otros.

Según la página oficial de OpenCV, su es libre y no tiene ningún costo, pues se distribuye bajo licencia BSD (Berkeley Software Distribution), lo cual indica uso libre parea múltiples proyectos académicos y comerciales, siempre y cuando, cumpla con las condiciones de la licencia. Es compatible con procesador tipo Intel, pero puede ser utilizada bajo cualquier otro tipo de procesadores y puede tomar ventaja de procesadores multinúcleo.

Las librerías OpenCV pueden instalarse tanto bajo Linux como bajo Windows. OpenCV cuenta con una buena eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. Está escrito en C y C++ optimizados, y se puede ejecutar en GNU/Linux, Windows, Android, iOS y Mac OS X. También es importante notar que hay

un activo desarrollo en las interfaces de Python, Ruby, Matlab, y otros lenguajes. OpenCV se estructura en cinco componentes principales (Yang, 2022)

- CV tiene algoritmos para procesar imágenes y de visión por ordenador en nivel básico y superior.
- ML es la biblioteca de aprendizaje de máquina.
- HighGUI contiene rutinas y funciones de I/O (entrada/salida) para el almacenamiento y carga de video e imágenes.
- CXCore contiene las estructuras básicas y algoritmos, apoyo XML, y funciones gráficas. Recibe información de CV, MIL y HighGUI.
- Archivo CvAux, que contiene algunos algoritmos experimentales.

3. Capítulo 3: Diseño metodológico

La metodología diseñada para el desarrollo del proyecto contempla las siguientes fases, las cuales se establecieron en función de los objetivos específicos definidos.

3.1.Fases metodológicas

Fase 1: Dimensionar

- Identificar las variables de entrada
- Estimar el tiempo y errores en el proceso actual de clasificación por color y apariencia en la finca el Cedro de Villa Garzón (Putumayo).
- Identificar métodos alternativos de medición de clasificación por color en la actualidad y seleccionar el apropiado de acuerdo con los requerimientos de diseño.
- Establecer los rangos de color y apariencia según para el proceso de clasificación.
- Determinar la cantidad de componentes que comprenden el sistema para procesamiento de imágenes
- Establecer el tipo de cámara, CPU y monitor necesario para el proceso de clasificación de chontaduro.

Fase 2: Diseño lógica de procesamiento de imágenes.

- Determinar las secuencias lógicas a implementar sobre el sistema de clasificación.
- Adquisición de las imágenes y construcción de las bases de datos.
- Diseño del sistema de clasificación en función del color y apariencia.

Fase 3: Evaluación

- Precisión: esta métrica determina la fracción de elementos clasificados correctamente como positivo entre todos los que el modelo ha clasificado como positivo.
- Recall: indica la proporción de ejemplos positivos que están identificadas correctamente por el modelo entre todos los positivos reales.
- F1-score: combina las métricas precisión y recall para dar un único resultado.

A continuación de detalla la metodología aplicada en el desarrollo del estudio.

3.2.Adquisición de imágenes proyecto

Para la adquisición de imágenes se empleó el modelo C505, es una cámara web con video HD 720p, en su calidad de video tiene componentes ópticos integrados en laptops con cámara web la cual ofrece video de calidad, colorido, fluido y nítido con formato diagonal de 60º, foco fijo y corrección de iluminación automática que se ajusta a la iluminación de cualquier espacio, otras características se definen en la tabla 2-1.

Tabla 2-1. Especificación técnica de la C505 HD

C505 HD WEBCAM									
Descripción	Resoluciones fps	Cámara visual	Enfoque automático	Corrección de iluminación	Micrófono Cancelación de ruido	conexión	cable	trípode	Tapa de privacidad
Cámara web HD con 720 y micrófono de largo alcance	HD 720/30 fps	60º	NO	RightLith2	Un micrófono omnidireccional de largo alcance	PLUG AND PLAY USB-A	2M	NO	NO

Fuente: Tomado y adaptado ficha técnica C505 HD, Autor (2023)

Figura 3-1. Camara C505 HD

Fuente: Tomado y adaptado ficha técnica C505 HD, Autor (2023)

Alguna de las características principales de esta componente en el proyecto es:

Figura 3-2. Características C505 HD

Característica	Descripción
Alta definición	Camara web HD con 720 p y micrófono de largo alcance con clara comunicación
Tipo conexión	Cable USB 2 mt
Video	calidad, colorido, fluido y nítido con formato diagonal de 60º, foco fijo y corrección de iluminación automática que se ajusta a la iluminación de cualquier espacio
Tipo Audio	omnidireccional posee tecnología de reducción de ruido
Dimensiones	Alto (31,91 mm), ancho (72,91 mm), profundidad (72,91 mm), peso (75 g)
Requisitos del sistema	-Computador de Windows 7 en adelante
	-macOS 10.10 en adelante
Especificaciones técnicas	Puerto USB-A
	Resolución máxima 720 p/30 fps
	Tipo de enfoque fijo
	Tipo de lente plástico
	Micrófono mono
	Cámara web con cable fijo USB-A de 2 mt

Fuente: Tomado y adaptado ficha técnica C505 HD, Autor (2023)

3.3.Montaje físico para simulación de procesamiento de imagines

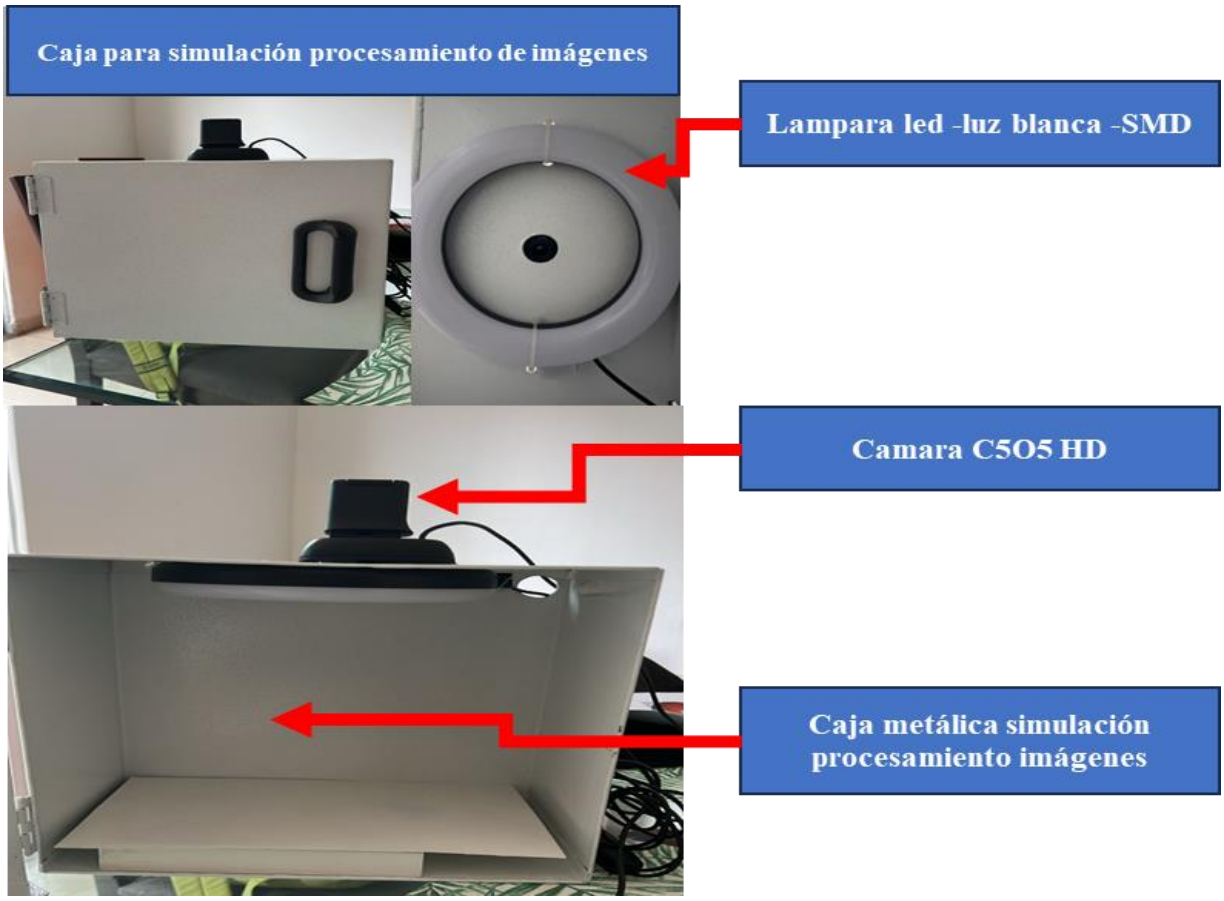
Para este proyecto se utilizó la cámara C5O5 HD descrita anteriormente, se utilizó una caja eléctrica la cual permite focalizar la luz del aro led al interior, creando condiciones de brillo y nitidez de las imágenes, las cuales son de 6 Pulgadas; la fuente de alimentación

del aro corresponde a la PC mediante conexión USB; las dimensiones de la caja metálica son 25 x 16 x 25 cm.

Con respecto a la luz led SMD esta tiene un rango de atenuación que va de 1% a 100%, emite luz blanca y tiene una potencia de 5W con fuente de alimentación de 5 V.

La conexión física para el desarrollo de pruebas se muestra a continuación tal cual como fue ensamblada para pruebas de procesamiento de imágenes.

Figura 3-3. Montaje Hardware para simulación procesamiento digital de imágenes

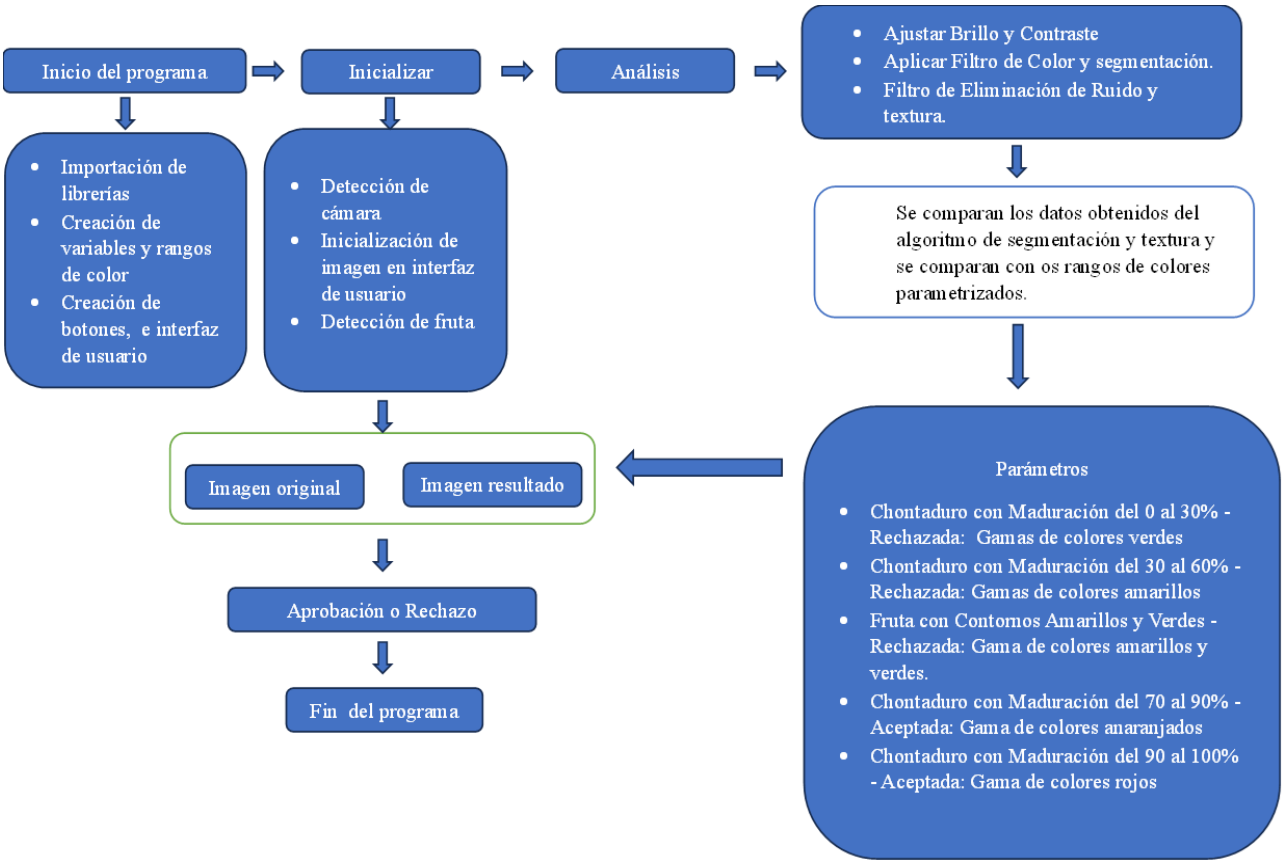


Fuente: Autor

4. Capítulo 4: Desarrollo de Hardware y software

En el presente capitulo se muestran el desarrollo tanto de hardware a través de códigos y librerías desarrolladas, así como los aspectos técnicos del programa como parámetros de color y apariencia como se describe a continuación y que opera según el siguiente diagrama

Figura 4-1. funcionamiento sistema de procesamiento imágenes



Fuente: Autor

4.1. Aspectos básicos de desarrollo proyecto

El desarrollo de este proyecto que implica la clasificación por color y apariencia del chontaduro para exportación se apoya en la visión por computadora y la programación en Python a través de una estructura controlada de iluminación por área.

Se implementó el procesamiento digital de imágenes asistida por computador a través de visión artificial (en nuestro caso cámara WEB C505 HD) debido a las siguientes razones:

- **Captura y análisis de datos visuales:** El procesamiento digital de imágenes permite capturar información visual, en este caso la clasificación de chontaduro por color y apariencia. Las imágenes proporcionan una fuente rica de datos visuales que son posibles convertirlos en datos numéricos esenciales para lograr este objetivo.
- **Precisión y objetividad:** Al utilizar algoritmos de procesamiento de imágenes, se puede lograr una clasificación precisa y objetiva de los chontaduros. A diferencia de la evaluación humana, que puede ser subjetiva, el procesamiento digital es coherente y no se ve influenciado por la fatiga, el sesgo o las variaciones en la percepción visual.
- **Eficiencia y escalabilidad:** Los algoritmos de procesamiento de imágenes pueden analizar grandes cantidades de imágenes en un corto período de tiempo. Esto es esencial para la clasificación de chontaduros destinados a la exportación, donde se pueden manejar grandes volúmenes de frutas de manera eficiente.
- **Automatización:** El procesamiento digital de imágenes permite la automatización de tareas que, de otra manera, serían tediosas y propensas a errores si se realizaran manualmente. La clasificación y el etiquetado de chontaduros se pueden realizar de manera continua y consistente con mínima intervención humana.
- **Capacidad para detectar características físicas específicas:** Los algoritmos de procesamiento de imágenes pueden ser diseñados para detectar características

específicas, como color, textura, forma y tamaño. Esto es fundamental para la clasificación por color y apariencia de los chontaduros.

- **Análisis de datos complejos:** Además de la clasificación, el procesamiento de imágenes permite realizar análisis más profundos, como la identificación de imperfecciones, la medición de la calidad y la detección de características físicas específicas. Esto puede ser fundamental para garantizar que los chontaduros cumplan con los estándares de exportación.
- **Consistencia y control de calidad:** La automatización y la objetividad del procesamiento de imágenes contribuyen a un mayor control de calidad. Se puede establecer criterios de clasificación y etiquetado precisos y consistentes que se adhieran a estándares específicos de calidad.
- **Capacidad de mejora continua:** Con el procesamiento digital de imágenes, puedes recopilar datos y retroalimentación constantes, lo que te permite mejorar y ajustar tus algoritmos con el tiempo. Esto es esencial para mantener y mejorar la precisión de la clasificación.

En resumen, el procesamiento digital de imágenes es la solución ideal para el objetivo principal porque te brinda la capacidad de capturar, analizar y clasificar datos visuales de manera precisa, eficiente y objetiva. Esto es fundamental en la clasificación de chontaduros destinados a la exportación, donde la calidad y la consistencia son críticas.

Para el procesamiento de dichas imágenes se utilizó el lenguaje de programación Python que no solo permite el análisis de dichas imágenes en datos, sino que además permite desarrollar software o aplicativos de interfaz humana permitiéndome generar un aplicativo funcional en cualquier sistema de cómputo o sistema operativo.

4.2. Aspectos básicos de desarrollo software

La elección de Python apoyado por librerías de OpenCV para el desarrollo del proyecto de detección y clasificación de chontaduros tiene varias razones fundamentales las cuales se describen a continuación:

- **Simplicidad y claridad del código:** Python es conocido por su sintaxis legible y fácil de entender. Esto es esencial, especialmente en proyectos de visión por computadora, donde la claridad del código es crucial. La simplicidad de Python facilita el desarrollo, la depuración y la colaboración en el proyecto.

- **Amplia comunidad y ecosistema:** Python cuenta con una comunidad de desarrolladores muy grande y activa, lo que significa que tienes acceso a una amplia gama de bibliotecas y herramientas de apoyo. OpenCV, por ejemplo, es una biblioteca ampliamente utilizada para el procesamiento de imágenes y visión por computadora en Python.

- **Flexibilidad y Versatilidad:** Python es un lenguaje altamente versátil que se utiliza en una variedad de campos, desde el análisis de datos hasta la inteligencia artificial. Esta versatilidad permite una fácil integración con otras herramientas y tecnologías, lo que puede ser beneficioso en proyectos complejos que requieren diversas funcionalidades.

- **Recursos Abundantes:** Existe una gran cantidad de recursos en línea, tutoriales y documentación disponible para Python y OpenCV. Esto simplifica el proceso de aprendizaje y resolución de problemas a medida.

- **Eficiencia en Procesamiento de Imágenes:** OpenCV es una biblioteca altamente eficiente para el procesamiento de imágenes y visión por computadora. Está

diseñada para trabajar con imágenes de manera rápida y precisa, lo que es fundamental en aplicaciones de detección y clasificación de objetos en tiempo real.

- **Desarrollo rápido:** Python es conocido por su rapidez en el desarrollo, ya que permite crear prototipos rápidamente y ajustar el código a medida que se avanza en el proyecto. Esto es especialmente útil cuando se requiere iterar y realizar mejoras en función de los resultados.

- **Interpretación y pruebas fáciles:** Python permite una fácil interpretación y pruebas interactivas, lo que facilita la exploración y experimentación con diferentes algoritmos y técnicas de procesamiento de imágenes. Esto es fundamental en proyectos de visión por computadora, donde a menudo se requieren ajustes y adaptaciones.

Python y OpenCV se eligen comúnmente en proyectos de visión por computadora debido a su facilidad de uso, eficiencia en el procesamiento de imágenes, amplia comunidad de soporte y recursos disponibles. Estas herramientas proporcionan una base sólida para el desarrollo exitoso de tu proyecto de detección y clasificación de chontaduro.

Otro punto importante en el desarrollo de este proyecto y cumplimiento de los objetivos de este es la iluminación y fondo en la captura y procesamiento de la imagen del chontaduro dado se debe tener un control de iluminación por área el proyecto por varias razones:

- **Consistencia de iluminación:** La iluminación controlada por áreas permite continuidad de esta en todas las imágenes que se capturan. Esto es fundamental para garantizar la consistencia en la apariencia de los chontaduros y en la detección de colores. La variabilidad en la iluminación puede introducir ruido en las imágenes y dificultar la precisión de la clasificación.

- **Eliminación de sombreado:** Al iluminar uniformemente el área de captura, se reduce o eliminan sombras no deseadas en las imágenes. Las sombras pueden distorsionar la apariencia de los objetos y dificultar la detección precisa de características, lo que es particularmente importante cuando se clasifican los chontaduros por apariencia.
- **Mejora de la claridad:** Una iluminación uniforme y controlada puede mejorar la claridad de las imágenes al resaltar los detalles y características de los chontaduros. Esto es esencial para el procesamiento de imágenes y la detección de imperfecciones, lo que contribuye a una clasificación precisa.
- **Reducción de Reflejos:** Controlar la iluminación puede ayudar a reducir los reflejos no deseados en la superficie de los chontaduros. Los reflejos pueden afectar negativamente la percepción del color y la apariencia, lo que es crítico en el proyecto.
- **Facilita la segmentación de objetos:** Una iluminación uniforme simplifica la tarea de segmentación de objetos. Esto es importante para separar con precisión los chontaduros del fondo y otros elementos en las imágenes.
- **Evita la influencia del entorno:** Al iluminar solo el área de captura, puedes minimizar la influencia de factores ambientales no controlados, como la iluminación natural o luces circundantes, que podrían afectar negativamente la consistencia de las imágenes.
- **Calibración más Sencilla:** Cuando la iluminación es constante y controlada, la calibración de los algoritmos de procesamiento de imágenes se vuelve más sencilla, ya que los valores de color y luminosidad se pueden definir con mayor precisión.
- **Optimiza la detección de color:** Para clasificar por color, una iluminación controlada garantiza que los colores sean representados de manera más precisa en las imágenes, lo que facilita la definición de rangos de colores y la clasificación.

- **Mejora la reproducibilidad:** La iluminación controlada facilita la repetición de las condiciones de captura, lo que es esencial para la reproducibilidad de los resultados y para realizar pruebas y ajustes de algoritmos con mayor precisión.
- **Menos ruido en imágenes:** Al eliminar variaciones no deseadas en la iluminación, se reduce el ruido en las imágenes, lo que contribuye a un procesamiento más preciso y resultados de clasificación más confiables.

En resumen, la iluminación controlada por áreas es esencial para garantizar la calidad y precisión en la captura de imágenes en la clasificación de chontaduros, ya que contribuye a la consistencia, claridad y reproducibilidad de estas, lo que es fundamental para el éxito del sistema de detección de color.

Para alcanzar y desarrollar los objetivos de plantea abarcar el problema desde dos puntos de vista, el primero incluye el parámetro de color y el segundo que corresponde a apariencia; para cualquiera de los dos casos se necesita empezar con la recolección de datos.

La recolección de datos es un paso crítico en el desarrollo del sistema de clasificación de color y apariencia de chontaduro para exportación. Para obtener un conjunto de datos representativo que incluya diferentes estados de madurez y calidad, así como diferentes gamas de color, se siguieron estos pasos detallados:

4.2.1. Diseño de la estrategia de recolección

Se tiene una base de datos de 80 imágenes de diferentes frutos de chontaduro en variedad de estados de maduración cosechados en la finca el Cedro de Villa Garzón (Putumayo).

1. **Obtención de imágenes**:

Fotografía propia: Toma de fotografías de chontaduros en diferentes estados de madurez y calidad teniendo en cuenta el nivel de iluminación y un fondo uniforme (blanco) para evitar que diferentes tipos de fondo en el fruto perturben el aspecto del color y posteriormente su análisis. Las imágenes fueron tomadas con un dispositivo móvil Motorola g60s quien cuenta con una cámara de 64 megapíxeles incluyendo un modo profesional.

2. Clasificación de gamas de color y madurez

La clasificación del chontaduro se realizará en base a los siguientes criterios:

Espacio de color HSV (Matiz, Saturación, Valor):

- **Matiz (H):** Representa el tono de color y varía de 0 a 360 grados. En el espacio de color HSV, los colores se representan en un círculo, donde los valores bajos (cerca de 0 y 360) corresponden a los rojos, y los valores más altos a los verdes y azules.
- **Saturación (S):** Representa la pureza del color y varía de 0 a 100%. Un valor bajo indica un color menos saturado, mientras que un valor alto indica un color más saturado.
- **Valor (V):** Representa la luminosidad o brillo del color y varía de 0 a 100%. Un valor bajo indica un color más oscuro, y un valor alto indica un color más claro.

Los rangos de colores se utilizan para definir las regiones de valores de matiz en el espacio de color HSV que corresponden a colores específicos que se desean identificar en la imagen. Cada rango especifica un rango de valores de matiz que define un color particular. Estos rangos se determinan de acuerdo con la apariencia de los objetos de interés en la imagen. Por ejemplo:

- (40, 80, 'Verde') define un rango de matiz que corresponde al verde.
- (20, 40, 'Amarillo') define un rango de matiz que corresponde al amarillo.
- (12, 20, 'Naranja') define un rango de matiz que corresponde al naranja.

- (0, 5, 'Rojo') y (175, 180, 'Rojo') definen dos rangos de matiz que corresponden al rojo. Nota que el rango de rojo se divide en dos partes debido a la naturaleza circular del espacio de color HSV.

Elección del espacio de color HSV: El espacio de color HSV se elige en lugar del espacio de color RGB porque es más intuitivo para describir colores por sus características perceptuales. Permite una mejor separación de los colores en función de su tono, saturación y brillo. Además, es más robusto ante cambios en la iluminación, lo que lo hace adecuado para aplicaciones de detección de colores en condiciones variables.

Los rangos de colores y el espacio de color HSV se utilizan para clasificar objetos en la imagen en función de sus colores. Esto es útil en aplicaciones como el análisis de la maduración de frutas u otros tipos de detección de colores en imágenes.

4.2.2. Clasificación de imágenes de chontaduro

Las imágenes del chontaduro se clasificaron en dos grandes categorías según su estado de madurez; estas son:

- **Maduración:** Presenta gamas de colores desde los colores amarillos y anaranjados.
- **Maduros:** Presenta el mayor porcentaje de gamas de colores rojos y en una menor proporción anaranjado. Se clasifican en buen estado y mal estado.

Se obtuvieron 20 imágenes de chontaduro para cada categoría, lo que resulta en un total de 80 imágenes en la base de datos.

Porcentaje de maduración:

- 'Rojo': '90-100%'
- 'Naranja': '70-90%'

- 'Amarillo': '30-60%'
- 'Verde': '0-30%'

Definición de Categorías

Para etiquetar la apariencia de los chontaduros, se deben definir categorías que reflejen aspectos relevantes de su estado físico. Estas categorías pueden incluir términos como "sin daños", "marcas superficiales", "deformaciones", entre otros. Cada categoría tiene el propósito de representar un estado específico de la fruta y proporcionar información valiosa sobre su calidad, para nuestro caso y objetivo principal se necesita chontaduros tipo exportación por lo que se tendrá como parámetro que el fruto este en un 95 % de perfección en cuanto marcas superficiales y deformaciones.

4.2.3. Implementación de Técnicas de Procesamiento de Imágenes

En este proyecto, Se implementan técnicas avanzadas de procesamiento de imágenes para etiquetar la apariencia de los chontaduros de manera más precisa y objetiva. Estas técnicas incluyen:

- **Detección de Bordes:** Utilizando algoritmos de detección de bordes, se pueden identificar áreas de transición en la superficie del chontaduro. Esto es valioso para detectar imperfecciones o irregularidades en su apariencia.
- **Corrección de Brillo y Contraste:** La corrección de brillo y contraste puede mejorar la visibilidad de los detalles en las imágenes, lo que facilita la detección de características importantes.
- **Segmentación:** La segmentación permite dividir la imagen en regiones que pueden corresponder a diferentes características del chontaduro, como zonas dañadas o deformadas.

- **Otras Técnicas de Procesamiento de Imágenes:** Además de las mencionadas, se pueden emplear diversas técnicas como filtrado, morfología matemática, extracción de características y análisis estadístico para evaluar la textura, forma y otras características visuales de los chontaduros.

Estas técnicas avanzadas permiten una evaluación más precisa y objetiva de la apariencia de los chontaduros, reduciendo la influencia de la subjetividad humana en el proceso de etiquetado.

4.3. Desarrollo de apariencia mediante Python

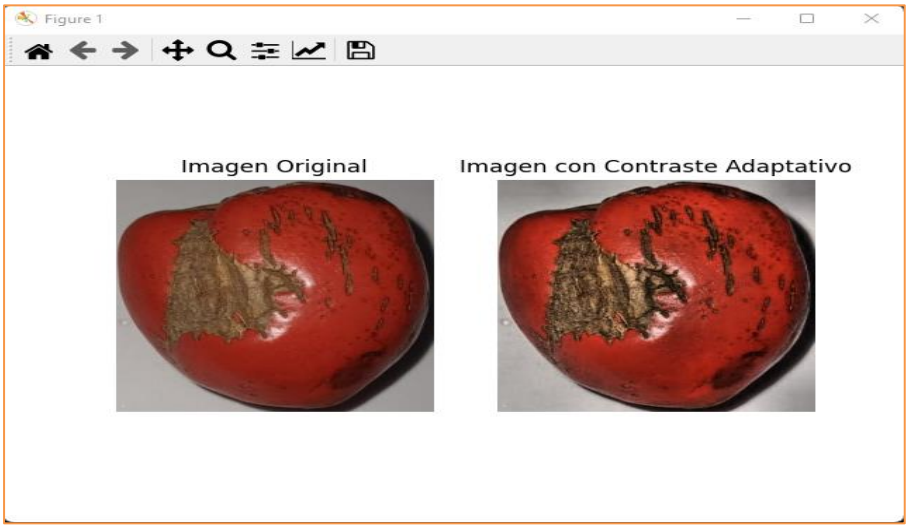
4.3.1. Corrección de brillo y contraste

El código empleado como parámetros de corrección y brillo comprenden la siguiente secuencia de programación:

```
from PIL import Image
import matplotlib.pyplot as plt
from skimage import exposure
import numpy as np
# Cargar la imagen
imagen = Image.open("prueba.jpeg")
# Convertir la imagen a una matriz NumPy
imagen_array = np.array(imagen)
# Aplicar una transformación de ajuste de contraste adaptativo
imagen_ajustada = exposure.equalize_adapthist(imagen_array)
# Mostrar solo la imagen original y la imagen ajustada de contraste adaptativo
plt.subplot(1, 2, 1)
plt.imshow(imagen_array)
plt.title('Imagen Original')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(imagen_ajustada)
plt.title('Imagen con Contraste Adaptativo')
plt.axis('off')
plt.show()
```

Según lo anterior al ingresar estos parámetros de código se obtienen los siguientes resultados:

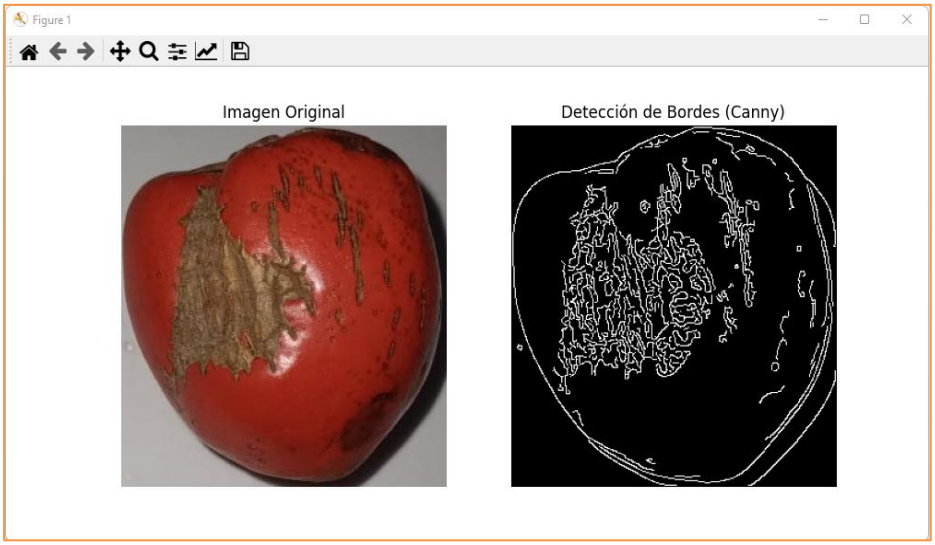
Figura 4-2. Ajuste brillo y apariencia



Fuente: Autor

A partir del código anterior se obtienen los siguientes resultados:

Figura 4-3. Ajustes bordes



Fuente: Autor

Para la determinación de rango de colores encontrados se establece el siguiente parámetro y para esta imagen en específico tenemos como respuesta:

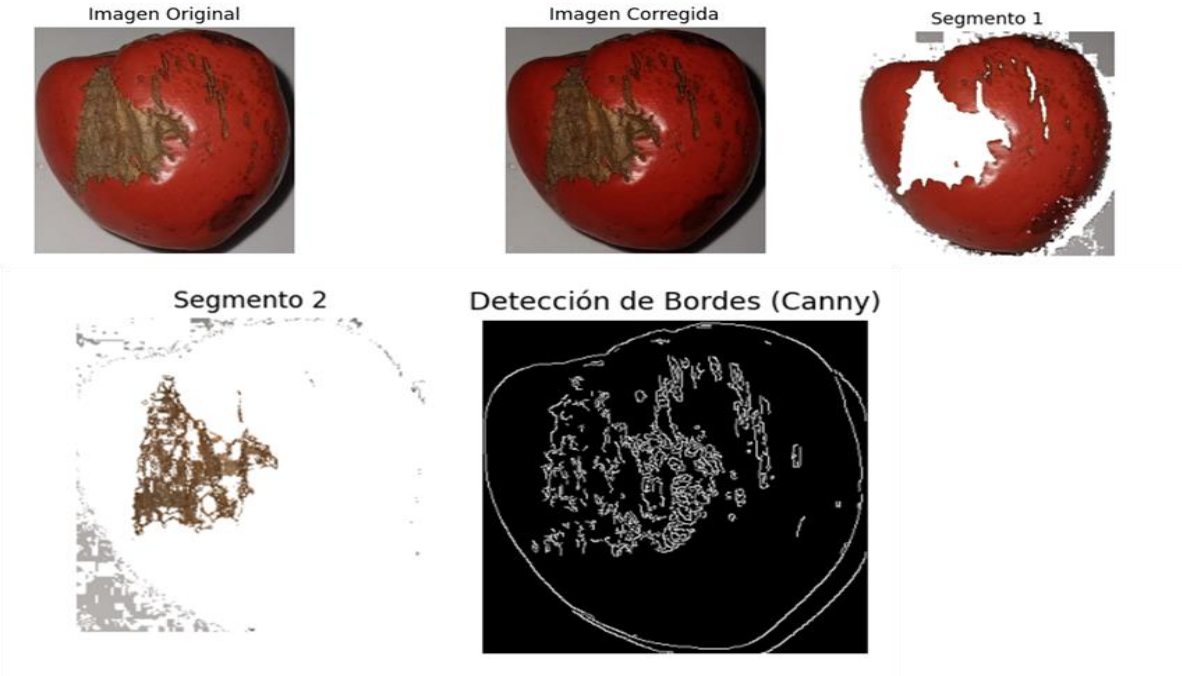
```
 # Definir el rango de colores 1
rango_bajo = np.array([1, 0, 0])
rango_alto = np.array([6, 255, 255])
# Definir el rango de colores 2
rango_bajo = np.array([10, 0, 0])
rango_alto = np.array([13, 255, 255])
```

Figura 4-4. Imagen de contraste y detección de bordes

Fuente: Autor

A partir de estos rangos se realiza una segmentación por medio de los parámetros encontrados, obteniendo los resultados de la figura 4-6.

Figura 4-5. Segmentación de imágenes

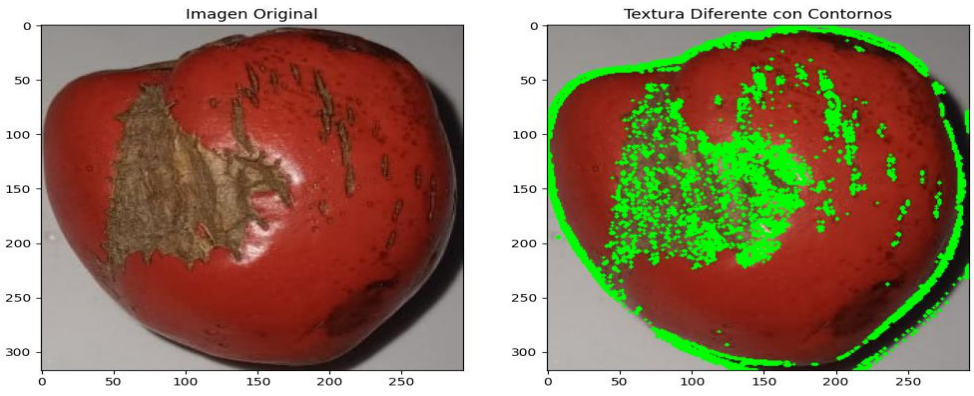


Fuente: Autor

4.4. Otras técnicas de procesamiento de imágenes

Para resaltar áreas de textura diferente en una imagen se aplica un filtro de diferencia de Gaussiano y luego se ajusta el umbral para destacar las áreas de textura que difieren significativamente, el código diseñado para este proceso es:

Figura 4-6. Imagen mediante Código Gaussiano



Fuente: Autor

Los anteriores algoritmos se utilizaron determinar la textura y apariencia del fruto de chontaduro luego más adelante se analiza e implementa si el área en píxeles de los segmentos encontrados de acuerdo con el análisis de rangos de color analizados en la figura, en correlación con el área de textura con contornos tienen una correlación de 95% se tiene una imperfección en la fruta por lo tanto no pasa la validación y se rechaza (Anexo 1 y 2).

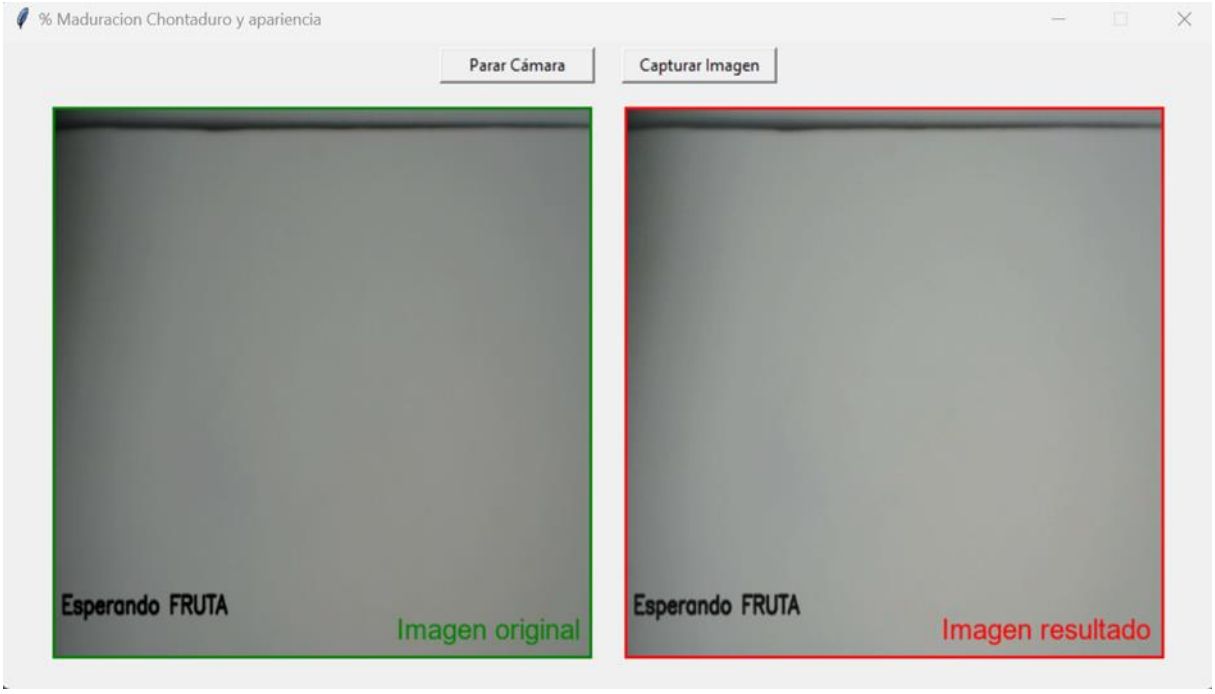
5. Capítulo 5: Resultados

Para la presentación de resultados se realizaron ensayos para determinar la respuesta del proceso de clasificación de chontaduro por procesamiento digital de imágenes con el fin de validar su funcionamiento y establecer algunos aspectos de diseño en caso de ser necesario; según lo anterior se tiene:

5.1. Iniciación y procesamiento en el programa

Partiendo de la base y mediante validación de ensayo que los componentes se encuentran correctamente conectados y operativos, el proceso de clasificación de chontaduro mediante condición simulada comprende:

Figura 5-1. Inicializando el programa y a la espera de fruta para analizar



Fuente: Autor

Una vez verificado que el sistema se encuentra operativo se procede a realizar ensayos con frutas de rangos de color verde amarillo en buen estado y mal estado con el fin

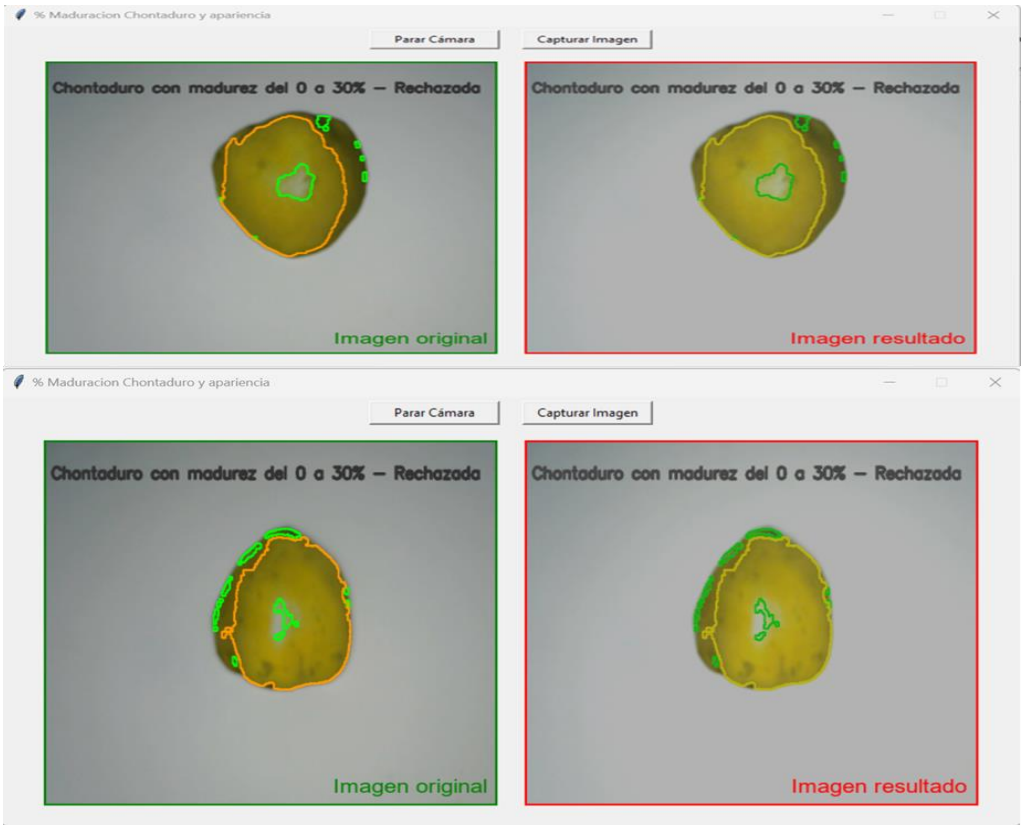
de validar la respuesta del sistema de procesamiento digital de imágenes, según lo anterior se tiene:

- Fruta amarilla buen estado.

El sistema determina que tiene rangos de color amarillos y verdes (delineado amarillo y verde) y aunque la fruta se encuentra en perfecto estado a nivel de apariencia este la rechaza por encontrarse en rango de no aprobación para el chontaduro tipo exportación.

Nota: El sistema siempre va a mostrar el menor % de maduración dado que es lo que se tiene en cuenta para determinar si entra o no en el rango de tipo exportación entre el 70 al 100 %) .

Figura 5-2. Ensayos de fruta verde

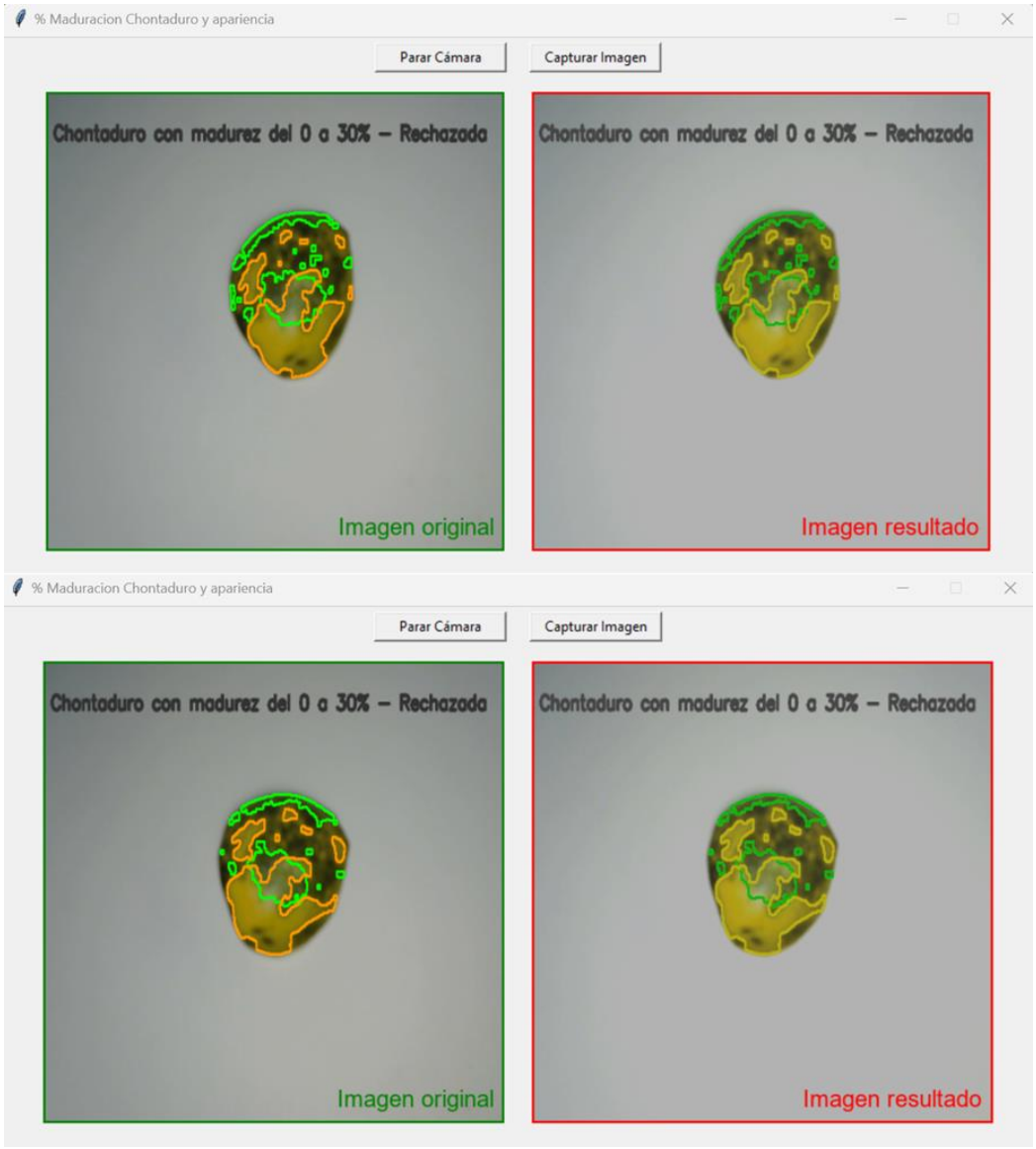


Fuente: Autor

- Fruta amarilla mal estado.

En este caso se puso a prueba de frutos cuya textura en algunos puntos era negra, permitiendo una lectura cuya notificación corresponde a chontaduro con madure del 0 al 30%-rechazado:

Figura 5-3. Ensayos con fruta verde con pigmentación negra

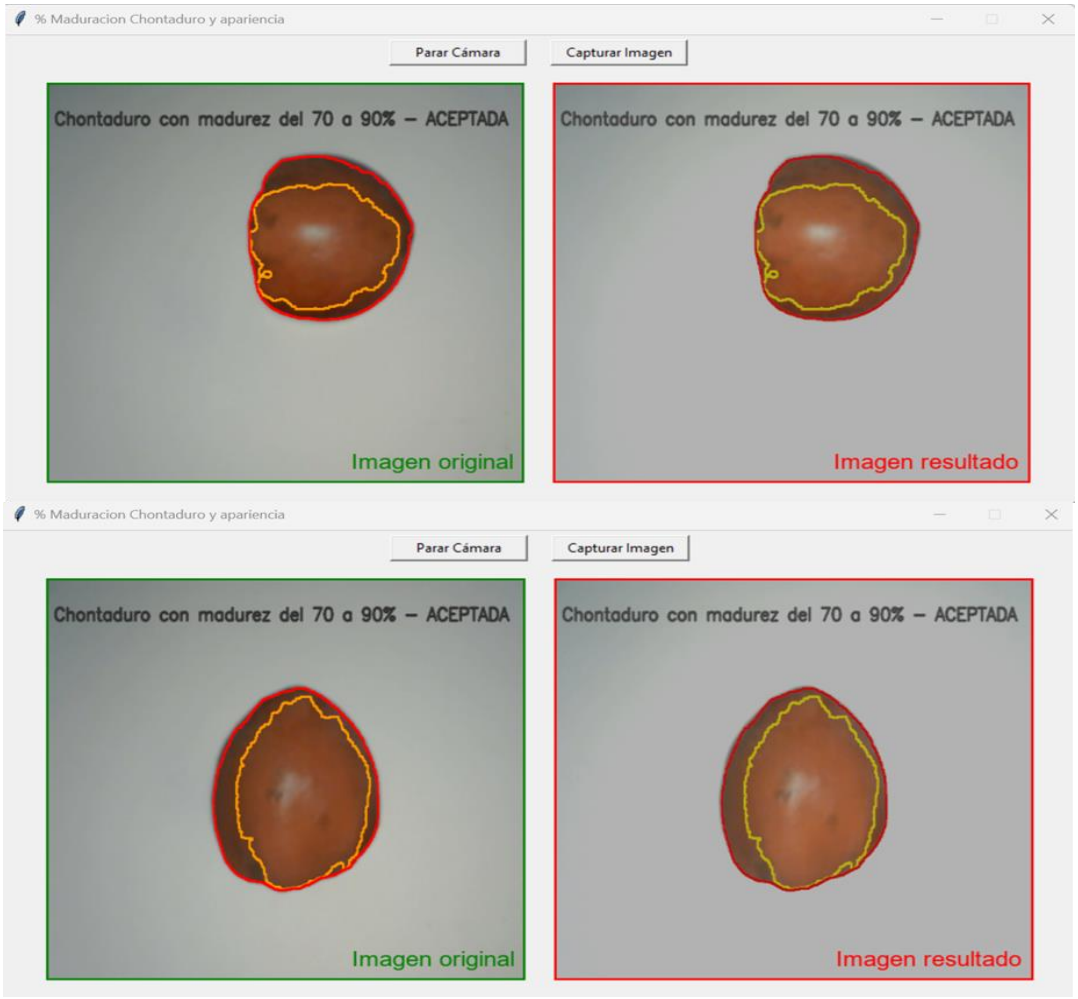


Fuente: Autor

- Fruta roja en buen estado

En este ensayo se tomaron de forma aleatoria 4 frutos con diferentes contornos y maduras (frutos rojos y anaranjados con apariencia buena); según lo anterior se tiene:

Figura 5-4. Ensayo fruta madura



Fuente: Autor

Al procesar la imagen de las frutas el sistema establece un contorno simétrico y define un nivel de madurez entre el 70 y 90%, dando como fruta aceptada en el proceso de clasificación. Este mismo proceso se repite con otras dos frutas indicando claramente los mismos resultados del proceso de clasificación como se indica en la figura

Figura 5-5. Ensayo 3 y 4 fruta madura

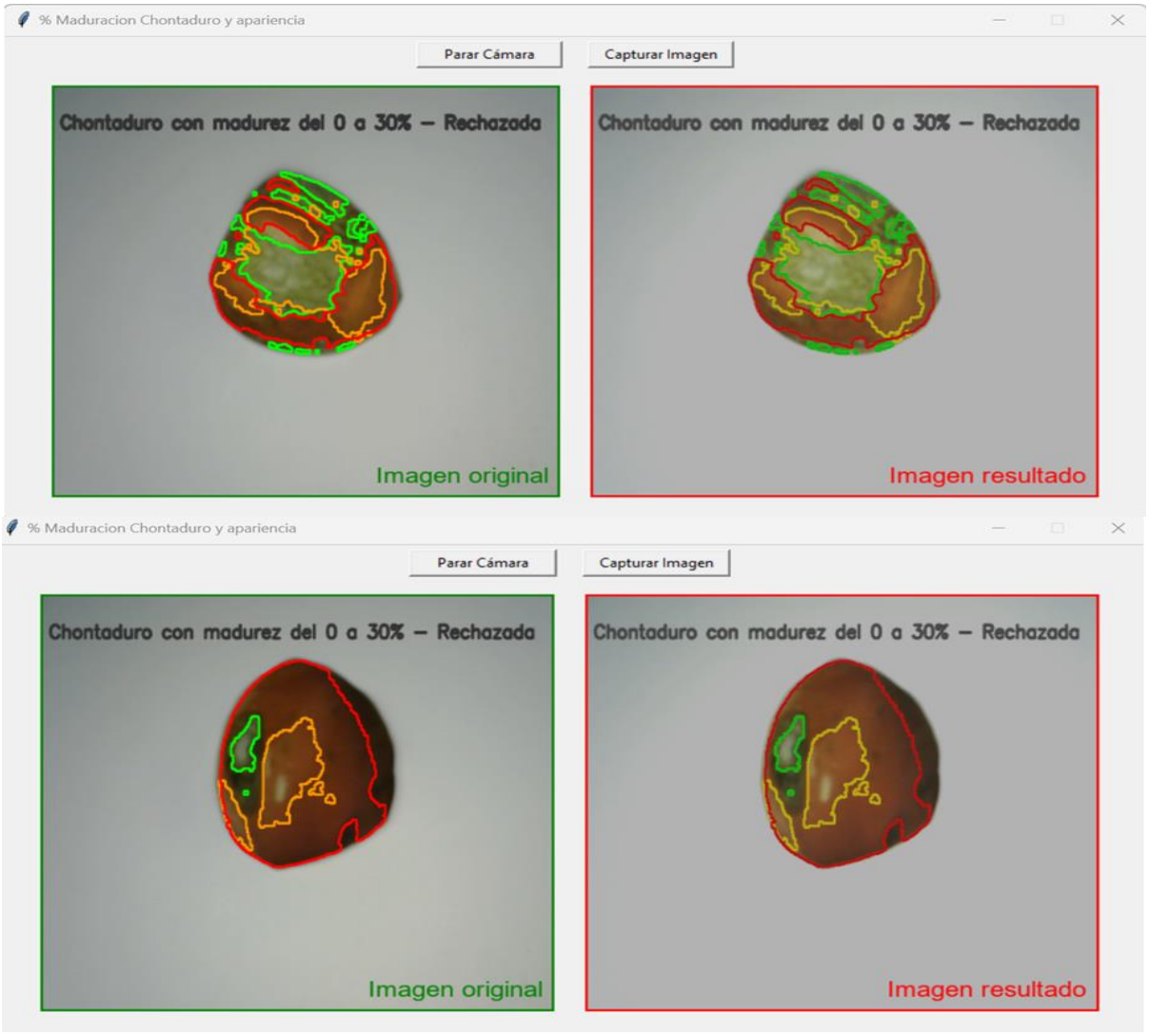
Fuente: Autor

Para validar el proceso de clasificación anterior se contrasta con frutas maduras dañadas las cuales presentan pigmentación negra con el fin de establecer el criterio de procesamiento de la imagen; a partir de lo anterior se tiene:

- Fruto maduro dañado

La fruta evaluada presenta un alto nivel de año, el cual al ser procesado indica chontaduro con madurez entre el 0 al 30% rechazada.

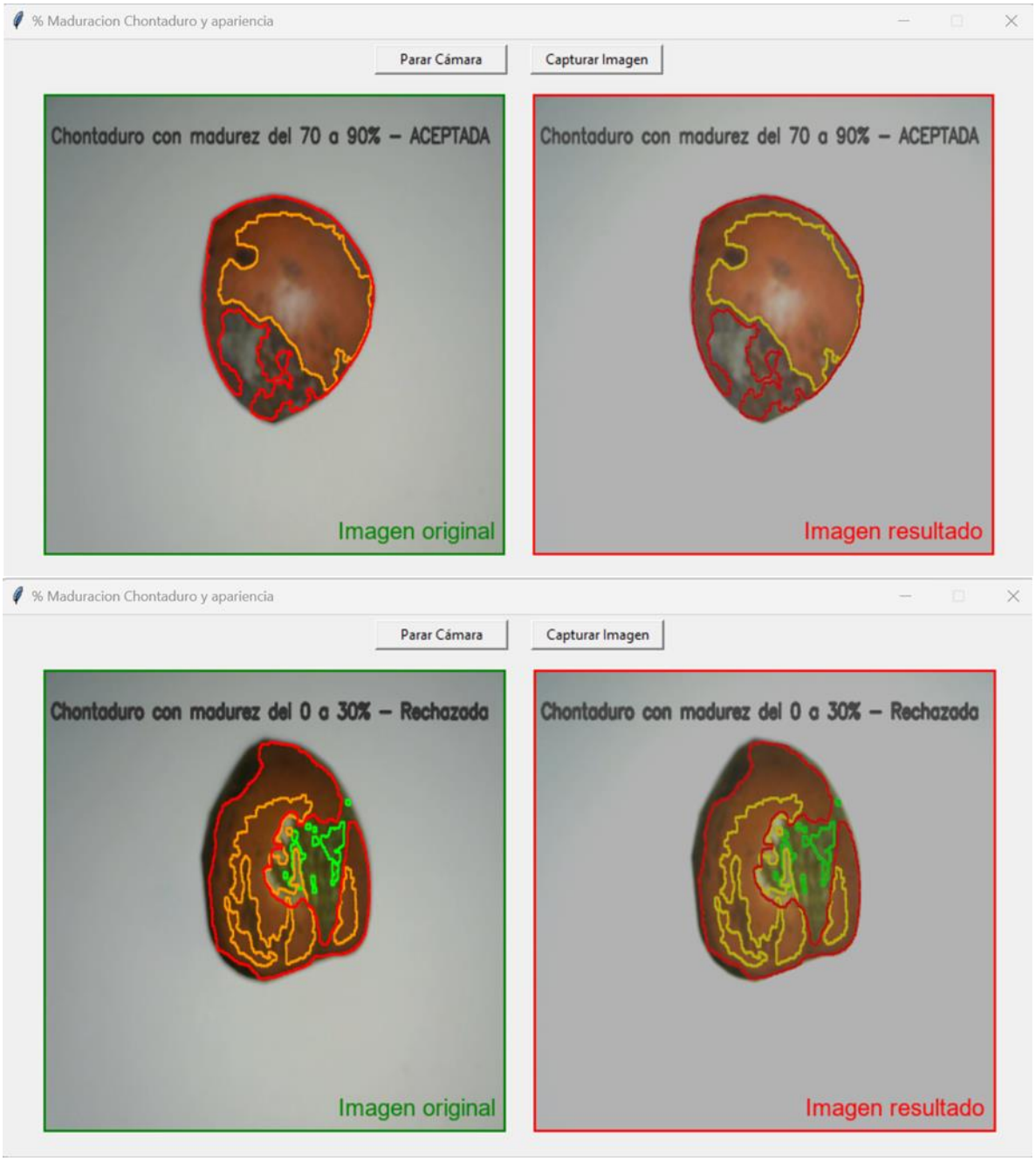
Figura 5-6. procesamiento de frutas dañadas



Fuente: Autor

Este mismo proceso se repite con los ensayos 3 y 4 de fruta madurada dañada

Figura 5-7. Ensayos 3 y 4 futa madura dañada



Fuente: Autor

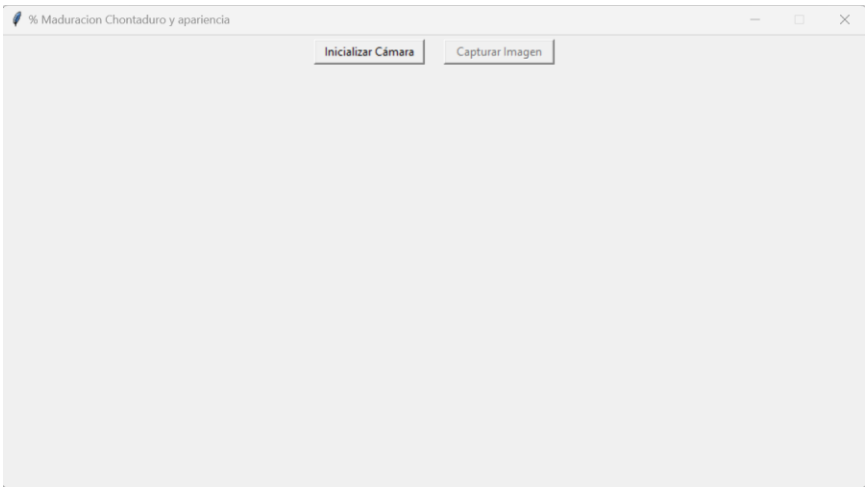
Un aspecto que tiene el sistema de procesamiento de imágenes se presenta cuando por error probable un operario puede accionar o dar clic al botón parar, aquí se apagan automáticamente la visualización de la imagen original y resultado para poder así mismo

cambiar entre muestra y muestra de fruto , de igual forma en caso de no utilizarse esta opción, el sistema funciona en tiempo real, es opcional al que está realizando la validación del fruto.

5.2.Almacenamiento de imágenes

El sistema posee un estado alterno para el procesamiento de imágenes que consiste en captura de imagen manual

Figura 5-8. Opción de capturar imagen



Fuente: Autor

Esta opción permite capturar la imagen cuando se le da clic en analizar y guardar en el directorio del equipo, la extensión de la imagen es “PNG” para poder después generar un informe más detallado de los hallazgos encontrados y analizados.

Una vez guardado con el nombre y ubicación que se desea , el sistema emite un mensaje informativo el cual es: La imagen se ha guardado exitosamente.

De los ensayos anteriores se ha validado la funcionalidad del sistema para el procesamiento de imágenes digitales para la clasificación de chontaduro, lo anterior permite

indicar adicionalmente que el proceso de color y apariencia para definir el estado de madurez y aceptabilidad es viable para dicha selección y posterior comercialización del producto.

5.3.Evaluación cuantitativa de tasa de aciertos

En el análisis de 80 chontaduros utilizando nuestro sistema de identificación, se logró una efectividad del 90%. Esto significa que el 90% de las frutas fueron correctamente categorizadas en términos de su estado y maduración, mientras que el 10% restante no pudo ser identificado con precisión.

El 10% de las chontaduros categorizados como 'No Identificado' representan los casos en los que nuestro sistema no pudo proporcionar una identificación confiable. Esto puede deberse a diversas razones, como la calidad de la imagen, la iluminación o la presencia de características y colores inusuales en los chontaduros.

Estos resultados indican que, si bien nuestro sistema es muy efectivo en la mayoría de los casos, existen escenarios en los que la identificación puede ser más desafiante. Esto nos brinda la oportunidad de mejorar aún más nuestro sistema para abordar situaciones en las que la identificación no es tan precisa, y así lograr una mayor efectividad en el futuro."

Tabla 5-1. Análisis cuantitativo

Chontaduro	Categoría	Estado	Identificación	Chontaduro	Categoría	Estado	Identificación
1	Maduración	Mal Estado	Identificación	41	Maduración	Buen Estado	Identificación
2	Maduración	Buen Estado	Identificación	42	Maduros	Mal Estado	Identificación
3	Maduración	Mal Estado	sin identificar	43	Maduros	Buen Estado	Identificación
4	Maduros	Buen Estado	Identificación	44	Maduración	Mal Estado	Identificación
5	Maduros	Mal Estado	Identificación	45	Maduros	Buen Estado	Identificación
6	Maduración	Mal Estado	Identificación	46	Maduros	Mal Estado	Identificación
7	Maduros	Mal Estado	Identificación	47	Maduración	Buen Estado	Identificación
8	Maduros	Buen Estado	sin identificar	48	Maduración	Buen Estado	Identificación
9	Maduros	Mal Estado	Identificación	49	Maduración	Mal Estado	Identificación
10	Maduros	Buen Estado	Identificación	50	Maduros	Buen Estado	Identificación
11	Maduros	Mal Estado	Identificación	51	Maduración	Buen Estado	Identificación
12	Maduros	Buen Estado	Identificación	52	Maduros	Buen Estado	Identificación
13	Maduración	Buen Estado	Identificación	53	Maduros	Buen Estado	Identificación
14	Maduros	Buen Estado	sin identificar	54	Maduros	Buen Estado	Identificación
15	Maduros	Buen Estado	Identificación	55	Maduros	Mal Estado	Identificación
16	Maduros	Buen Estado	Identificación	56	Maduros	Mal Estado	Identificación
17	Maduración	Buen Estado	Identificación	57	Maduros	Mal Estado	sin identificar
18	Maduración	Buen Estado	Identificación	58	Maduros	Mal Estado	Identificación
19	Maduros	Buen Estado	Identificación	59	Maduración	Mal Estado	Identificación
20	Maduración	Mal Estado	Identificación	60	Maduración	Mal Estado	Identificación
Chontaduro	**Categoría**	**Estado**		**Chontaduro**	**Categoría**	**Estado**	
21	Maduración	Buen Estado	Identificación	61	Maduración	Buen Estado	Identificación
22	Maduración	Mal Estado	sin identificar	62	Maduración	Mal Estado	Identificación
23	Maduros	Buen Estado	Identificación	63	Maduros	Mal Estado	Identificación
24	Maduración	Mal Estado	Identificación	64	Maduros	Buen Estado	Identificación
25	Maduros	Buen Estado	Identificación	65	Maduros	Buen Estado	Identificación
26	Maduros	Mal Estado	Identificación	66	Maduros	Mal Estado	Identificación
27	Maduración	Mal Estado	Identificación	67	Maduros	Mal Estado	Identificación
28	Maduros	Buen Estado	Identificación	68	Maduros	Buen Estado	sin identificar
29	Maduros	Mal Estado	Identificación	69	Maduración	Mal Estado	Identificación
30	Maduración	Mal Estado	Identificación	70	Maduración	Mal Estado	Identificación
31	Maduros	Buen Estado	Identificación	71	Maduros	Buen Estado	Identificación
32	Maduración	Buen Estado	Identificación	72	Maduración	Mal Estado	Identificación
33	Maduros	Mal Estado	Identificación	73	Maduración	Buen Estado	Identificación
34	Maduración	Buen Estado	sin identificar	74	Maduros	Buen Estado	Identificación
35	Maduración	Mal Estado	Identificación	75	Maduración	Buen Estado	Identificación
36	Maduros	Mal Estado	Identificación	76	Maduración	Buen Estado	Identificación
37	Maduros	Mal Estado	sin identificar	77	Maduros	Mal Estado	Identificación
38	Maduros	Buen Estado	Identificación	78	Maduración	Buen Estado	Identificación
39	Maduros	Buen Estado	Identificación	79	Maduración	Mal Estado	Identificación
40	Maduración	Buen Estado	Identificación	80	Maduración	Buen Estado	Identificación

Fuente: Autor

Conclusiones

El desarrollo e infraestructura del proyecto demostró ser adaptable y flexible a las necesidades planteadas dado que determina de manera precisa el % de maduración del fruto de chontaduro en función de su gama de clores presentando de manera delineada el contorno de parámetros de color encontrados, teniendo como base que los colores rojos presentan un porcentaje de maduración de 90 - 100%, los colores naranja de 70 - 90%, amarillos presentan un porcentaje de maduración de 30 - 60% y colores verdes presentan un porcentaje de maduración de 0 - 30%. Destacando que en cuanto a apariencia el sistema determina que las áreas encontradas y no delineadas son de apariencia oscura rechazando la aprobaron del fruto así presente las gamas de colores adecuados que para nuestro caso son colores anaranjados a rojos siempre y cuando no presenten texturas diferentes en la fruta.

En cuanto al sistema implementado y desarrollado en Python y OpenCV, demostró ser eficiente en la clasificación de chontaduro. La combinación de técnicas de segmentación, análisis de color y filtro de textura permitió identificar áreas de la fruta que cumplen con los parámetros de color deseados, al tiempo que excluye las partes dañadas. Este proyecto se considera y presenta como una solución viable y prometedora para la clasificación de chontaduro en la finca el Cedro de Villa Garzón. Su implementación podría mejorar la eficiencia del proceso de selección y clasificación de la fruta en la finca.

La investigación ha demostrado la relevancia de la aplicación de técnicas de procesamiento de imágenes y análisis de color en la clasificación de productos agrícolas. Esta tecnología tiene el potencial de reducir el desperdicio, ya que la fruta clasificada cuenta con la calidad y exigencia del cliente y del mercado, lo que evita devoluciones y costos, a su vez aumenta la competitividad en el mercado de exportación apoyado así en el análisis de técnicas de fertilización e implementación de pesticidas en la planta del fruto, ya que se puede identificar aspectos fitosanitarios y focalizar intervenciones en sitio, así mismo poder determinar la rentabilidad del producto en funciones de producción de calidad para exportación.

En cuanto a a tasa de aciertos en el procesamiento digital, según análisis cuantitativo a una muestra de 80 chontaduros se logró establecer un porcentaje de aciertos del 90% y de desaciertos del 10 %.

Se pueden implementar sistemas de desarrollo y análisis más complejos como es el aprendizaje profundo de redes neuronales siempre, teniendo en cuenta que para esto se necesita una considerable base de datos que alimente este método y su eficacia tienda a la mayor precisión posible.

Anexos

A. Anexo 1. Determinación de rangos de color mediante código Gaussiano

```
 import cv2
import numpy as np
# Cargar la imagen original
imagen_original = cv2.imread('prueba.jpeg')
# Convertir la imagen a espacio de color HSV
imagen_hsv = cv2.cvtColor(imagen_original, cv2.COLOR_BGR2HSV)
# Definir los rangos de colores y sus respectivas categorías
rangos_de_colores = [(40, 80, 'Verde'), (20, 40, 'Amarillo'), (12, 20, 'Naranja'), (0, 5, 'Rojo'), (175, 180,
'Rojo')]
# Inicializar una lista para almacenar información de cada región
regiones = []
# Recorrer cada región
for rango in rangos_de_colores:
    rango_bajo = np.array([rango[0], 0, 0])
    rango_alto = np.array([rango[1], 255, 255])
    mascara = cv2.inRange(imagen_hsv, rango_bajo, rango_alto)
        # Calcular el área de la región
    area = cv2.countNonZero(mascara)
        # Calcular el histograma de colores en la región
    hist_hue_region = cv2.calcHist([imagen_hsv], [0], mascara, [180], [0, 180])
        # Encontrar el valor más alto en el histograma (color predominante)
    color_predominante = np.argmax(hist_hue_region)
        regiones.append({
        'rango': rango,
        'area': area,
        'color_predominante': color_predominante
    })

# Encontrar la región con el área más grande
region_mas_grande = max(regiones, key=lambda x: x['area'])
# Encontrar el color con mayor predominancia
colores_predominantes = [region['color_predominante'] for region in regiones]
color_mas_predominante = max(set(colores_predominantes), key=colores_predominantes.count)
# Interpretar los resultados
color_interpretado = None
for rango in rangos_de_colores:
    if rango[0] <= color_mas_predominante <= rango[1]:
        color_interpretado = rango[2]
if color_interpretado:
    print(f'El área de la región más grande es: {region_mas_grande["area"]} píxeles')
    print(f'El color predominante en la imagen es: {color_interpretado}')
    # Definir los porcentajes de maduración según el color
    porcentaje_maduracion = {
        'Rojo': '90-100%',
        'Naranja': '70-90%',
        'Amarillo': '30-60%',
        'Verde': '0-30%'
    }
    # Mostrar el porcentaje de maduración
    print(f'El fruto se encuentra entre un {porcentaje_maduracion[color_interpretado]} de maduración')
else:
    print(f'El color predominante en la imagen no se encuentra en los rangos especificados.')
```

Respuesta

El área de la región más grande es: 54855 píxeles

El color predominante en la imagen es: Rojo

El fruto se encuentra entre un 90-100% de maduración

Anexo 2. Implementación de todo el sistema

```
import cv2              #Libreria de open cv

from tkinter import *   #Libreria de tinker (bibloteca grafica)
from tkinter import filedialog, messagebox     #Abrir archivos y mostrar
from PIL import Image, ImageTk, ImageDraw, ImageFont     #imagenes
import numpy as np      #bibloteca numerica

 # Variables para almacenar los rangos de colores
# Amarillo
Amarillo_min = np.array([30, 50, 50], np.uint8)
Amarillo_max = np.array([30, 255, 255], np.uint8)
verde_claro_min  = np.array([30, 50, 50], np.uint8)   #Parametro de verde claro minimo en HSV
verde_claro_max = np.array([100, 255, 255], np.uint8)   #Parametro de verde claro maximo en HSV
# Naranja
naranja_min = np.array([3, 100, 100], np.uint8)    #Parametro de naranja minimo en HSV
naranja_max = np.array([50, 255, 255], np.uint8)   #Parametro de naranja maximo en HSV
# Rojo oscuro
rojo_oscuro_min = np.array([0, 50, 50], np.uint8)       # OTROS RANGOS(10 50 50) Parametro rojo
oscuro minimo en HSV
rojo_oscuro_max = np.array([20, 255, 255], np.uint8)     # OTROS RANGOS(30 255 255) Parametro
rojo oscuro maximo en HSV
#Rojo': 90-100%
#Naranja: 70-90%
#Amarillo': 30-60%
#Verde: 0-30%
class CameraApp:

    def __init__(self, root):
        self.root = root   #Abrir camara
        self.root.title("% Maduracion Chontaduro y apariencia")  #Titulo de la ventana en otros metodos
de la clase
        screen_width = self.root.winfo_screenwidth()   #ancho de la imagen
        screen_height = self.root.winfo_screenheight() #Altura de la imagen
        x = (screen_width - 900) // 2  #Posicion horizontal -900 ancho de la ventana
        y = (screen_height - 600) // 2   #Posicion vertical -600 altura deseada de la ventana
        self.root.geometry(f"900x470+{x}+{y}")    #Tamaño y posicion de imagen
        self.root.resizable(False, False)         #
            self.camera = cv2.VideoCapture(0)     #Inicio de camara
        self.camera_initialized = False
        self.camera_active = False
        self.frame_top = Frame(root)         #Organizar y obtener otros widtegs
        self.frame_top.pack(pady=5)          #Agrega espacio de 5 pixeles
            self.initialize_button = Button(self.frame_top, text="Inicializar Cámara",
command=self.toggle_camera, width=15)  #Inicio de boton de inicializar camara
        self.initialize_button.pack(side=LEFT, padx=10)

        self.capture_button = Button(self.frame_top, text="Capturar Imagen",
command=self.capture_image, width=15, )    #Boton de captura de pantalla
        self.capture_button.pack(side=LEFT, padx=10)
```

```
        self.video_frame = Frame(root, width=900, height=400)  #widget tiene un ancho de 900 pixeles y
un alto de 400 pixeles
        self.video_frame.pack()
        self.video_label_original = Label(self.video_frame)  #
        self.video_label_original.pack(side=LEFT, padx=10, pady=10)
        self.video_label_modified = Label(self.video_frame)
        self.video_label_modified.pack(side=LEFT, padx=10, pady=10)
        self.color_mean = np.array([0, 0, 0], dtype=np.int32)
        self.area = 0
        self.frame_count = 0
        self.color_sum = np.array([0, 0, 0], dtype=np.int32)
        self.area_sum = 0
        self.initialize_camera()
         def toggle_camera(self):
        if self.camera_active:
            self.stop_camera()
        else:
            self.initialize_camera()
    def initialize_camera(self):
        if not self.camera_active:
            self.camera_active = True
            self.initialize_button.config(text="Parar Cámara")
            self.capture_button.config(state=NORMAL)
            self.show_camera()
    def stop_camera(self):
        if self.camera_active:
            self.camera_active = False
            self.initialize_button.config(text="Inicializar Cámara")
            self.capture_button.config(state=DISABLED)
            self.video_label_original.configure(image='')
            self.video_label_modified.configure(image='')
    def show_camera(self):
        if self.camera_active:
            _, frame = self.camera.read()
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame_hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)  # Convertir de RGB a HSV

            kernel = np.ones((5, 5), np.uint8)

           # determina si la forma contine máximos y mínimos verde claro % de maduracion de 0 a 30%
            mask_verde_claro = cv2.inRange(frame_hsv, verde_claro_min, verde_claro_max)
            mask_verde_claro = cv2.erode(mask_verde_claro, kernel, iterations=1)
            mask_verde_claro = cv2.dilate(mask_verde_claro, kernel, iterations=1)
            contours_verde_claro, _ = cv2.findContours(mask_verde_claro, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            cv2.drawContours(frame, contours_verde_claro, -1, (0, 255, 0), 2)

           # determina si la forma contine máximos y mínimos amarillo % de maduracion de 30 a 70%
            mask_amarillo = cv2.inRange(frame_hsv, Amarillo_min, Amarillo_max)
            mask_amarillo = cv2.erode(mask_amarillo, kernel, iterations=1)
            mask_amarillo = cv2.dilate(mask_amarillo, kernel, iterations=1)
```

```
        contours_amarillo, _ = cv2.findContours(mask_amarillo, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cv2.drawContours(frame, contours_amarillo, -1, (0, 255, 255), 2)

                # determina si la forma contine Rangos máximos y mínimos naranja % de maduracion
de 70 a 90%
        mask_naranja = cv2.inRange(frame_hsv, naranja_min, naranja_max)
        mask_naranja = cv2.erode(mask_naranja, kernel, iterations=1)
        mask_naranja = cv2.dilate(mask_naranja, kernel, iterations=1)
        contours_naranja, _ = cv2.findContours(mask_naranja, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cv2.drawContours(frame, contours_naranja, -1, (255, 154, 0), 2)

       # determina si la forma contine Rangos máximos y mínimos ROJO % de maduracion de 90 a
100%
                  mask_rojo_oscuro = cv2.inRange(frame_hsv, rojo_oscuro_min, rojo_oscuro_max)
        mask_rojo_oscuro = cv2.erode(mask_rojo_oscuro, kernel, iterations=1)
        mask_rojo_oscuro = cv2.dilate(mask_rojo_oscuro, kernel, iterations=1)
        contours_rojo_oscuro, _ = cv2.findContours(mask_rojo_oscuro, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cv2.drawContours(frame, contours_rojo_oscuro, -1, (255, 0, 0), 2)


        # Verificar si hay contornos de color verde claro
        if len(contours_verde_claro) > 0:
          for contour in contours_verde_claro:
          # Calcular centro del contorno
           M = cv2.moments(contour)
           cx = int(M['m10'] / M['m00'])
           cy = int(M['m01'] / M['m00'])
           # Verificar si centro del contorno está dentro del rango de color de verde claro
           if cv2.pointPolygonTest(contour, (cx, cy), False) >= 0:
              print("Chontaduro con maduración de 0 a 30%")
              font = cv2.FONT_HERSHEY_SIMPLEX
              text_position = (10, 50)
              text = "Chontaduro con madurez del 0 a 30% - Rechazada  "
              cv2.putText(frame, text, text_position, font, 0.72, (50, 50, 50), 2, cv2.LINE_AA)

        # Verificar si hay contornos de color AMARILLO
        elif len(contours_amarillo) > 0:
          for contour in contours_naranja:
          # Calcular centro del contorno
            M = cv2.moments(contour)
            cx = int(M['m10'] / M['m00'])
            cy = int(M['m01'] / M['m00'])
            # Verificar si centro del contorno está dentro del rango de color de naranja
            if cv2.pointPolygonTest(contour, (cx, cy), False) >= 0:
              print("Chontaduro con maduración de 30 a 60%")
              font = cv2.FONT_HERSHEY_SIMPLEX
              text_position = (10, 50)
              text = "Chontaduro con madurez del 30 a 60% - Rechazada  "
              cv2.putText(frame, text, text_position, font, 0.72, (50, 50, 50), 2, cv2.LINE_AA)
```

```
    # Verificar si hay contornos amarillos y verdes, independientemente de otros colores
    elif len(contours_amarillo) > 0 and len(contours_verde_claro) > 0:
        print("Fruta con contornos amarillos y verdes - Rechazada")
        font = cv2.FONT_HERSHEY_SIMPLEX
        text_position = (10, 150)
        text = "Fruta con contornos amarillos y verdes - Rechazada"
        cv2.putText(frame, text, text_position, font, 0.72, (50, 50, 50), 2, cv2.LINE_AA)

    # Verificar si hay contornos de color naranja
    elif len(contours_naranja) > 0:
      for contour in contours_naranja:
      # Calcular centro del contorno
        M = cv2.moments(contour)
        cx = int(M['m10'] / M['m00'])
        cy = int(M['m01'] / M['m00'])
        # Verificar si centro del contorno está dentro del rango de color de naranja
        if cv2.pointPolygonTest(contour, (cx, cy), False) >= 0:
          print("Chontaduro con maduración de 70 a 90%")
          font = cv2.FONT_HERSHEY_SIMPLEX
          text_position = (10, 50)
          text = "Chontaduro con madurez del 70 a 90% - ACEPTADA"
          cv2.putText(frame, text, text_position, font, 0.72, (50, 50, 50), 2, cv2.LINE_AA)

    # Verificar si hay contornos de color rojo oscuro
    elif len(contours_rojo_oscuro) > 0:
      for contour in contours_rojo_oscuro:
      # Calcular centro del contorno
        M = cv2.moments(contour)
        cx = int(M['m10'] / M['m00'])
        cy = int(M['m01'] / M['m00'])
        # Verificar si centro del contorno está dentro del rango de
        if cv2.pointPolygonTest(contour, (cx, cy), False) >= 0:
          print("Chontaduro con maduración de 90 a 100%")
          font = cv2.FONT_HERSHEY_SIMPLEX
          text_position = (10, 100)
          text = "Chontaduro con madurez del 90 a 100% - ACEPTADA"
          cv2.putText(frame, text, text_position, font, 0.72, (50, 50, 50), 2, cv2.LINE_AA)

    else:
        font = cv2.FONT_HERSHEY_SIMPLEX
        text = "Esperando FRUTA"
        text_position = (10, 450 - 10)
        cv2.putText(frame, text, text_position, font, 0.72, (0, 0, 0), 2, cv2.LINE_AA)  # Color negro

  # Redimensionar y mostrar la imagen original en la etiqueta correspondiente
    frame = cv2.resize(frame, (400, 400))
    image_original = Image.fromarray(frame)
    draw_original = ImageDraw.Draw(image_original)
    font_original = ImageFont.truetype("arial.ttf", 20)  # Cambia el tamaño de fuente aquí
    text_original = "Imagen original"
```

```
        text_width_original, text_height_original = draw_original.textsize(text_original,
font=font_original)
        text_position_original = (255, 400 - text_height_original - 10)
        draw_original.text(text_position_original, text_original, fill="GREEN", font=font_original)
        # Dibujar un enmarcado rojo en toda la imagen
        draw_original.rectangle([0, 0, 399, 399], outline="GREEN", width=2)

        # Mostrar la imagen original en la etiqueta correspondiente
        image_original = ImageTk.PhotoImage(image_original)
        self.video_label_original.configure(image=image_original)
        self.video_label_original.image = image_original


        # Aplicar filtros de imagen a la imagen modificada
        filtered_frame = self.adjust_brightness_contrast(frame)
        filtered_frame = self.apply_color_overlay(filtered_frame)
        filtered_frame = self.apply_.apply_denoising_filter(filtered_frame)


        # Redimensionar y mostrar la imagen modificada en la etiqueta correspondiente
        filtered_frame = cv2.resize(filtered_frame, (400, 400))
        image_modified = Image.fromarray(filtered_frame)
        draw_modified = ImageDraw.Draw(image_modified)
        font_modified = ImageFont.truetype("arial.ttf", 20)  # Cambia el tamaño de fuente aquí
        text_modified = "Imagen resultado"
        text_width_modified, text_height_modified = draw_modified.textsize(text_modified,
font=font_modified)
        text_position_modified = (400 - text_width_modified - 10, 400 - text_height_modified - 10)
        draw_modified.text(text_position_modified, text_modified, fill="RED", font=font_modified)
        # Dibujar un enmarcado rojo en toda la imagen
        draw_modified.rectangle([0, 0, 399, 399], outline="red", width=2)


        # Mostrar la imagen modificada en la etiqueta correspondiente
        image_modified = ImageTk.PhotoImage(image_modified)
        self.video_label_modified.configure(image=image_modified)
        self.video_label_modified.image = image_modified

        self.root.after(10, self.show_camera)

    def adjust_brightness_contrast(self, frame): #filtro o ajuste de brillo
        alpha = 1.5  # Ajuste de brillo
        beta = 25  # Ajuste de contraste
        adjusted_frame = cv2.convertScaleAbs(frame, alpha=alpha, beta=beta)
        return adjusted_frame

    def apply_color_overlay(self, frame): # filtro de color segmentacion
        color_overlay = np.full_like(frame, self.color_mean, dtype=np.uint8)
        filtered_frame = cv2.addWeighted(frame, 0.7, color_overlay, 0.3, 0)
        return filtered_frame

    def apply_denoising_filter(self, frame): # filtro de textura
```

```
        denoised_frame = cv2.fastNlMeansDenoisingColored(frame, None, 1, 1, 1, 1) # parametros a
modificar
        return denoised_frame

    def capture_image(self):
        _, frame = self.camera.read()
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(frame)

        save_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG Image",
"*.png")])
        if save_path:
            try:
                image.save(save_path)
                messagebox.showinfo("Guardado exitoso", "La imagen se ha guardado correctamente.")
            except Exception as e:
                messagebox.showerror("Error al guardar", f"No se pudo guardar la imagen. Error: {str(e)}")

    def __del__(self):
        self.stop_camera()

root = Tk()
app = CameraApp(root)
root.mainloop()
```

Referencias Bibliográficas

Arévalo, J. (2018). *La librería de visión artificial OPENCV.* Málaga (España): Universidad de Málaga. Obtenido de http://mapir.isa.uma.es/varevalo/drafts/arevalo2004lva1.pdf

Chamorro, I. (2018). *Cultivo y agroindustria de palmito de chontaduro : ¿una alternativa de desarrollo local?. Estudio de caso - Asociación de productores agropecuarios el Cuembí APAC, en Puerto Asís, Putumayo.* Bogotá: Universidad Pontificia Javeriana. Obtenido de http://hdl.handle.net/10554/34198

CORPOICA. (2021). *El cultivo de Chontaduro para fruto y palmito.* Bogotá: CORPOICA. Obtenido de https://repository.agrosavia.co/bitstream/handle/20.500.12324/15990/39971_24526.pdf?sequence=1&isAllowed=y

Daza, A. (2015). Cambios fisiológicos, texturales el chontaduro. *Biotecnología en el Sector Agropecuario y Agroindustrial, 13*(2), 67-75. Obtenido de http://www.scielo.org.co/pdf/bsaa/v13n2/v13n2a08.pdf

FINAGRO. (12 de Abril de 2023). Obtenido de https://www.finagro.com.co/sites/default/files/chontaduro_putumayo_0.pdf

García, J. (2015). Uso del procesamiento de imágenes digitales para medir los parámetros morfométricos de partículas. *Revista científica Técnica, 1*(1), 14-44. Obtenido de https://www.redalyc.org/pdf/2230/223040405003.pdf

Mejía, R. (2014). *Procesamiento Digital.* Mexico D.F.: UASLP. Obtenido de http://laurence.com.ar/artes/comun/Apuntes%20procesamiento%20digital%20de%20imagenes.pdf

Menon, K. (2021). Digital grading and sorting of fruits. *MaterialsToday: Proceedings, 43*(6), 3749-3758. doi:https://doi.org/10.1016/j.matpr.2020.10.989

Muñoz, E. (2021). *Diseño de una máquina para la clasificación de tomate chonto.* Bogotá: Universidad de la Salle. Obtenido de https://ciencia.lasalle.edu.co/cgi/viewcontent.cgi?article=1781&context=ing_automatizacion

Patil, P. (2021). Grading and sorting technique of dragon fruits using machine learning algorithms. *Journal of Agriculture and Food Research, 4*(1), 230-283. doi:DOI:10.1016/j.jafr.2021.100118

Pourdarbani, R. (2018). Study on an automatic sorting system for Date fruits. *Journal of the Saudi Society of Agricultural Sciences*(14), 83-90. Obtenido de https://doi.org/10.1016/j.jssas.2013.08.006

Ramírez, W. (2017). Determinación del estado de madurez de una cereza aplicando procesamiento de imágenes. *Revista de Jóvenes en la ciencia, 3*(2), 2685-2689. Obtenido de 2112-Texto%20del%20artículo-7111-1-10-20171215.pdf

Reyes, J. (2021). *Load Cell and Arduino Technology as Lemon Fruit Classifier.* Matti (Filipinas): Science direct. Obtenido de ORIGINES, DOMINGO JR and DELOS REYES, JAN RIC, Load Cell and Arduino Technology as Lemon Fruit Classifier (February 26, 2021). Available at SSRN: https://ssrn.com/abstract=3793375 or http://dx.doi.org/10.2139/ssrn.3793375

Rodríguez, J. (2020). *Diseño, construcción y automatización de un prototipo de máquina clasificadora de piñas de acuerdo al color y forma por medio de visión artificial.* Riobamba: ESPC. Obtenido de http://dspace.espoch.edu.ec/bitstream/123456789/14314/1/15T00740.pdf

Serrano, m. (2018). *Segmentación de imágenes.* Cataluña (España): Universidad Politécnica de Catalunya. Obtenido de https://dialnet.unirioja.es/servlet/tesis?codigo=240859

Serrano, T. (2018). *Scene Context Classification with Event Driven Spiking Deep Neural Networks.* Sevilla (España): Instituto de Microelectronica de Sevilla. Obtenido de https://idus.us.es/bitstream/handle/11441/102356/Scene%20Context%20Classification%20with%20Event-Driven%20Spiking.pdf?sequence=1&isAllowed=y

Torres, J. (2022). *Informe de producción agrícola y sostenible 2022.* Mocoa (Putumayo): UNTS.

UNC. (1 de Mayo de 2023). Obtenido de https://www.famaf.unc.edu.ar/~pperez1/manuales/cim/cap2.html

Yang, Z. (2022). Audit Optimization Based on Computer Vision Technology. *Journal of Computer and Communications , 10*(1), 50-58. doi:DOI: 10.4236/jcc.2022.1010004