UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN MECATRÓNICA

TEMA:

"***Sistema para determinar la falta de nutrientes a través de las hojas en el cultivo de mango.***"

AUTOR: DAVID ISRAEL HERNÁNDEZ HERNÁNDEZ

DIRECTOR: MSC. IVÁN IGLESIAS NAVARRO, ING.

ASESOR: ING, XAVIER LIMA TRUJILLO, ING.

ASESOR: PHD, DAVID OJEDA PEÑA, ING.

IBARRA – ECUADOR

2022

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

El cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte.

DATOS DEL AUTOR			
CÉDULA DE IDENTIDAD:	1004134498		
APELLIDOS Y NOMBRES:	Hernández Hernández David Israel		
DIRECCIÓN:	Río Cenepa 120, Río Lita		
EMAIL:	dihernandezh@utn.edu.ec		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0987542085
DATOS DE LA OBRA			
TÍTULO:	"Sistema para determinar la falta de nutrientes a través de las hojas en el cultivo de mango"		
AUTOR:	Hernández Hernández David Israel		
FECHA (DD/MM/AAAA):	28/09/2022		
SOLO PARA TRABAJOS DE GRADO			
PROGRAMA:	Pregrado		
TÍTULO POR EL QUE OPTA:	Ingeniero en Mecatrónica		
ASESOR/DIRECTOR:	MSc Iván Iglesias, Ing.		

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y la desarrollo sin violar derechos de autores de terceros, por lo tanto, la obra es original, y que es titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de esta y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, 28 de septiembre de 2022

Nombre: David Israel Hernández Hernández

Cedula: 1004134498

CERTIFICACIÓN

En calidad de tutor del presente Trabajo de Grado titulado: "Sistema para determinar la falta de nutrientes a través de las hojas en el cultivo de mango", certifico que fue desarrollado por el señor David Israel Hernández Hernández, bajo mi supervisión.

IVAN IGLESIAS NAVARRO

Firmado digitalmente porIVAN IGLESIAS NAVARRO
Motivo:Apruebo este documento
Fecha:2022-09-27 14:43-05:00

MSc. Iván Iglesias, Ing.

DIRECTOR DEL PROYECTO

DEDICATORIA

A mis padres por haberme forjado como la persona que soy en la actualidad. Muchos de mis logros se los debo a ustedes entre los que se incluye este. Su formación de reglas y algunas libertades me motivaron siempre para conseguir mis anhelos.

AGRADECIMIENTO

El principal agradecimiento a mi familia y a mi novia por su comprensión y estímulo constante, además su apoyo incondicional a lo largo de esta carrera.

A todas los maestros, compañeros y personas que de una y otra forma me apoyaron en la realización de este trabajo.

RESUMEN

En este trabajo de investigación se muestra el proceso de diseño y programación de un sistema de detección de deficiencias nutricionales, el cual está realizado con el lenguaje de programación Python y permite al usuario conocer el estado de su planta a partir de una fotografía capturada en un ambiente controlado.

Las deficiencias nutricionales presentes durante el cultivo de mango son de vital importancia durante todo el proceso que conlleva su producción. Para el productor es difícil conocer cuando su cultivo está sufriendo estas deficiencias por lo que es necesario un análisis foliar para conocer su estado.

El sistema desarrollado es capaz de reconocer cada hoja y realizar el análisis respectivo a cada una de ellas de acuerdo a la deficiencia que el usuario desee buscar teniendo en cuenta que a partir de matrices de confusión se obtuvo que el algoritmo de detección de deficiencia de nitrógeno tiene una eficacia del 74,2% y una especificidad del 96,5%.; el algoritmo de detección de deficiencia de fósforo tiene un 73,7% para eficacia y 78,3% de especificidad; mientras que el algoritmo de detección de potasio tiene 64,1% para eficacia y 85,9% de especificidad.

Palabras clave: *visión artificial, visión por computadora, Python, OpenCV, análisis foliar, mango, deficiencias nutricionales.*

ABSTRACT

This research work shows the design and programming process of a nutritional deficiency detection system, which is carried out with the Python programming language and allows the user to know the status of their plant from a photograph captured in a controlled environment.

The nutritional deficiencies present during mango cultivation are of vital importance throughout the process that entails its production. For the producer it is difficult to know when his crop is suffering from these deficiencies, so a foliar analysis is necessary to know its status.

The developed system is capable of recognizing each leaf and carrying out the respective analysis of each one of them according to the deficiency that the user wishes to search for, taking into account that from confusion matrices it was obtained that the nitrogen deficiency detection algorithm it has an efficacy of 74.2% and a specificity of 96.5%; the phosphorus deficiency detection algorithm has 73.7% for efficacy and 78.3% for specificity; while the potassium detection algorithm has 64.1% efficiency and 85.9% specificity.

Keywords: *artificial vision, computer vision, Python, OpenCV, foliar analysis, mango, nutritional deficiencies.*

Índice General

Índice de figuras

Índice de tablas

Índice de ecuaciones

INTRODUCCIÓN

Problema

Basándose en el cambio de matriz productiva del ecuador, es necesaria una reducción en las importaciones y un incremento de exportaciones, el cual se deriva del aumento de producción nacional de productos procesados o materia primera, en este caso la producción de frutas exóticas como materia prima[1].

En el ecuador se deben adoptar nuevas tecnologías para maximizar sus beneficios. Esto no ocurre, mayormente, por la falta de conocimiento de los agricultores o por no tener una cultura o noción técnica de su aplicación en sus cultivos. Esta innovación permite aplicar de manera específica de acuerdo a las necesidades de cada región, mediante el procesamiento de imágenes, éstas se digitalizan, geo-referencian, y de acuerdo a sus características se puede establecer la proporción de abono a emplear en cada punto exacto del terreno.

De acuerdo a fuentes consultadas de información agrícola, en el 2016 existían unas 18.000 hectáreas (ha) dedicadas a este cultivo, con una producción estimada de 82.246 toneladas. Para el año 2019 existe una superficie cosechada de alrededor de 20.000 ha, con una producción estimada de 170.109 toneladas[2].

En el caso de Imbabura los sectores y áreas dedicadas al cultivo de mango se encuentran en el cantón Ibarra (específicamente en la parroquia rural Ambuquí) y en el cantón Pimampiro (parroquia urbana Pimampiro) con un total de 197 hectáreas dedicadas al cultivo de esta fruta, y los costos de producción varían de acuerdo al tamaño del productor, predominando los medianos productores.

El árbol de mango dulce, puede ser bastante tolerante a diferentes condiciones de fertilidad, pero siempre es necesario un programa de fertilización para que de esta manera se obtengan rendimientos requeridos para frutas de excelente calidad[3].

Dentro de los fertilizantes más importantes durante el cultivo se encuentran el nitrógeno, fosforo y potasio, cada uno con diferentes aportaciones hacia el cultivo de entre los cuales los se destaca que aportan al crecimiento, formación, de raíces, yemas florales, promueve el almacenamiento de nutrientes y azúcar, entre otros[4].

El agricultor no puede identificar fácilmente si su cultivo se encuentra de buena salud, por lo que necesita un sistema fácil de usar en el que sea capaz de identificar el estado de su cultivo[5].

Además, se deben adoptar nuevas tecnologías para maximizar sus beneficios. Esto no ocurre, mayormente, por la falta de conocimiento de los agricultores o por no tener una cultura o noción técnica de su aplicación en sus cultivos; sin saber las ventajas que puede proporcionar la aplicación de tecnología y nuevos sistemas de producción, como la agricultura de precisión[6].

La visión artificial por computadora es una disciplina que siempre se encuentra en crecimiento ya que existen múltiples aplicaciones en cualquier área de tecnología. Con los avances de la tecnología, se busca facilitar y mejorar la calidad de vida de las personas con discapacidad visual, usando dispositivos que utilicen las nuevas tecnologías. [7]La visión por computador con sus distintos algoritmos de procesamiento podría ser sustituto de la visión humana, ya que funciona como herramienta fundamental en la implementación de dispositivos de apoyo a personas con discapacidades, especialmente de aspecto visual[8].

De esta manera, se presenta el desarrollo sistema para la detección de deficiencias de nutrientes usando visión artificial[9].

Objetivos

Objetivo General

- Diseñar un sistema para determinar la falta de nutrientes principales a través de las hojas en el cultivo de mango por medio de visión artificial.

Objetivos Específicos

- Identificar las principales características de las hojas del mango para relacionar con sus deficiencias nutricionales.
- Determinar los algoritmos de visión artificial para el procesamiento de imágenes.
- Implementar el algoritmo de visión artificial en el sistema para la detección de las deficiencias nutricionales.
- Validar su funcionamiento.

Alcance

Este trabajo de grado tiene la finalidad de diseñar un sistema capaz de reconocer una las deficiencias presentes en las hojas de una planta de mango, mediante el uso de visión artificial. Las hojas de esta fruta se reconocerán por su color por medio de una cámara de gama media en un ambiente medianamente controlado, seguidamente se llevarán las imágenes al computador donde serán procesadas para la deficiencia de nutrientes presente en las mismas.

Justificación

La presente investigación se enfocará en el estudio de herramientas tecnológicas, como, visión artificial, la cual permita reconocer las hojas, en este caso, de la planta del mango de azúcar, para conocer la deficiencia de nutrientes presente en la planta. Esto se realiza con el motivo de optimizar el tiempo que un agricultor emplea realizando la actividad de revisar los cultivos, y de alguna manera reducir gastos dependiendo de los obreros que sean utilizados en esta tarea. Así el agricultor obtendrá un producto de mayor calidad optimizando tiempo y

recursos. Gracias a la visión artificial a través de un algoritmo que sea capaz que procesar las imágenes obtenidas.

CAPÍTULO I

MARCO TEÓRICO

Como se ha detallado en el problema, el objetivo general del presente trabajo de grado es detectar las deficiencias nutricionales de nitrógeno, fosforo y potasio, en la planta del mango a través de sus hojas. Esto se realizará por medio de visión artificial en un sistema aplicativo. Para esto es importante que los términos y conceptos sean claros.

1.1. Mango

En la actualidad, excluyendo al banano, el mango es el tercer producto tropical más popular mundialmente, antecedido de la piña y el aguacate. El volumen total de las exportaciones de mango ha aumentado en los últimos, por estadísticas de la FAO, en 2014 existía una producción de casi 80.000 toneladas, mientras que en el 2016 la producción llegaba a 170.000 toneladas. Por lo que su mercado es de gran representación para nuestro país[3]. Ver figura 1-1.

Figura 1-1

Fruto de mango a poco tiempo de ser cosechado.

1.2. Hoja de Mango

Las hojas son simples, sin estípulas, alternas, con pecíolos de no más de 12 centímetros de largo. Su forma y tamaño llegan a ser variables, pero generalmente su largo es mucho mayor que su ancho, con extremos redondeados a afilados. Su tamaño varía según el árbol. Su parte superior es brillante y la inferior tiene un verde claro, mientras que en su totalidad la tonalidad es un verde oscuro [10]. Ver figura 1-2.

Dependiendo de la edad de la planta o árbol de mango, estas cambian su color de verde claro a marrón o de púrpura a verde oscuro. El color variable de la hoja joven se puede utilizar como rasgo distintivo entre variedades. Se debe tomar en cuenta que, si lo nutrientes son escasos en el suelo, ocurren variaciones en su color[11].

Figura 1-2.

Hoja de mango sana o sin ninguna deficiencia.

1.3. Nutrientes NPK en el cultivo de mango

Es importante tener un buen programa de fertilización para el crecimiento en la plantación[12]. De esta manera se obtiene que los nutrientes principales del mango son el nitrógeno (N), fosforo (P) y potasio (K)[13].

1.3.1. Nitrógeno.

Su principal función es brindar mayor vegetación y crecimiento ya que estimula el desarrollo de las yemas florales[11].

1.3.2. Fósforo.

Se encarga de la formación de raíces y aceleración de formación de frutos. Además, una vez formado el fruto, es parte de los componentes energéticos[10].

1.3.3. Potasio

Su función consiste en aumentar la resistencia a enfermedades y plagas, además de transportar foto asimilados al fruto y sitios de reserva[14].

1.4. Consecuencias Por Falta De Nutrientes

La falta o deficiencia de nutrientes en la hoja de mango tienen diversas consecuencias que se ven reflejadas en el fruto y en la hoja[15]. Los cuales se mencionan a continuación:

1.4.1. Nitrógeno

Su deficiencia afecta al crecimiento de plantas jóvenes, además provoca raquitismo y la producción se ve afectada en cantidad. Sus hojas se vuelven de un verde pálido que evoluciona hasta un amarillo muy fuerte[5].

1.4.2. Fósforo

Su deficiencia tiene lugar con una sequía precoz en sus ramas y queda prematura de las hojas. Como producto la hoja tendrá una necrosis en su ápice por lo que en un inicio la hoja se torna de un color verde oscuro que evoluciona hasta un morado rojizo, hasta la muerte de esa parte de la hoja [16].

1.4.3. Potasio

La planta se vuelve poco resistente a enfermedades y plagas por lo que de igual manera su producción se ve afectada considerablemente. Los primeros síntomas de su carencia se presentan en los bordes de las hojas con un color naranja, o amarillo rojizo.

Una vez se conocen todas las características de la hoja de mango con sus respectivas deficiencias principales y características visuales principales se presenta una tabla con la información relevante obtenida[17].

1.5. Análisis foliar en el cultivo de mango

El análisis de foliar en el cultivo de mango sirve para diagnosticar y para confirmar diagnósticos basados en síntomas visuales de deficiencia o toxicidad[18].

El muestreo debe ser realizado como mínimo un 2% del área que ocupe cada zona de muestreo. El muestreo de hojas se debe realizar antes de la poda y después de una cosecha. Las hojas de los árboles seleccionados deben tomarse a una altura correspondiente a la mitad de la copa, y cada hoja corresponderá a un punto cardinal, es decir, se muestrean 4 hojas por cada árbol[13].

A partir de estas características se tiene como resultado una tabla con las principales características visuales que se presentan cuando existe deficiencias NPK. Ver tabla 1.

Tabla 1

Características visuales de deficiencias nutricionales

Deficiencias	Características visuales principales
Nitrógeno	Su principal característica se manifiesta como clorosis es decir la presencia de un color verde pálido y amarillo en la totalidad de la hoja, por lo general, ocupando más del 80% de su totalidad. [5]
Fósforo	Se presenta como necrosis en el ápice o punta superior de la hoja. La necrosis provocada produce coloraciones que varían entre marrón, rojo y hasta purpura[16]
Potasio	Se presenta como necrosis en los lados de la hoja con colores representativos como el marrón o amarillo intenso[17].

A medida que la tecnología crece, las nuevas innovaciones sustituyen a la vieja maquinaria. La combinación de la visión por computadora con la robótica formuló un nuevo enfoque innovador para la agricultura a través de estudios periódicos de la tierra e inspección de datos. Las innovaciones recientes en el manejo de cultivos en sitios específicos incluyen drones, sistemas aéreos no tripulados para imágenes de cultivos[9].

1.6. Visión Artificial

Se trata de conceptos como la imitación del sistema visual humano a través de diversos conocimientos sobre cómo funcionan los esquemas de la cámara, las proyecciones y la fotogrametría[7]. En pocas décadas, la visión por computadora hizo su existencia combinada en todos los campos posibles, como el reconocimiento de patrones, el aprendizaje automático, los gráficos por computadora, las reconstrucciones en 3d, la realidad virtual y la realidad aumentada.[19] En 2010, la visión por computadora pudo realizar tareas de alto nivel como el reconocimiento de objetos, la navegación autónoma de vehículos, la detección de rostros, el reconocimiento de huellas dactilares, el procesamiento rápido de imágenes por computación y la navegación robótica.[8]

Técnicas como la detección de líneas, la extracción de características, las segmentaciones, la coincidencia y el seguimiento de características, la optimización y la reconstrucción de la realidad 3D abrieron las puertas a una gran cantidad de invenciones asombrosas, como la localización y el mapeo visual simultáneo (slam), el seguimiento de objetos, etc.[20]

1.6.1. Procesamiento de imágenes

El procesamiento de imágenes consiste en la manipulación de imágenes, incluida la superresolución, eliminación de ruido, eliminación de neblina, eliminación de brillo, etc. [21]Existe una variedad de métodos basados en el aprendizaje profundo propuestos para realizar en uno o varios tipos de tareas de procesamiento de imágenes. [22]A diferencia de los métodos anteriores, exploramos la capacidad de los grandes modelos y el gran volumen de datos. Luego, se introduce un modelo de preentrenamiento que maneja varias tareas de procesamiento de imágenes.[23]

1.6.2. Lenguaje de programación orientado a la visión artificial

Python es un lenguaje de programación orientado a objetos de alto nivel, es un lenguaje de código abierto simple e intuitivo. La velocidad de Python se ha extendido por todo el mundo, es el resultado continuo de miles de programadores, es fácil de obtener, fácil de instalar y tiene muchas llamadas al sistema y bibliotecas. Además, se puede utilizar como lenguaje de extensión para aplicaciones que requieren una interfaz programable.[24]

1.6.3. Librería OpenCV

Son las siglas de *open source computer vision library*, que consta de más de 300 funciones. Puede funcionar en computadoras con *Windows* o *Linux*. *OpenCV* es una forma de postularse para la comunidad de visión de código abierto que es muy útil en la oportunidad de actualizar la aplicación de la visión por computadora en línea con el crecimiento de la computadora

personal que continúa creciendo. [25]Este software proporciona una serie de funciones de procesamiento de imágenes, así como funciones de análisis de imágenes y patrones.[26]

Algunos ejemplos de aplicaciones de *OpenCV* son identificación, segmentación y reconocimiento de objetos; como detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en el video, rastrear movimientos de cámara, rastrear objetos en movimiento. *OpenCV* es una biblioteca muy conocida en el procesamiento de imágenes de visión por computadora.[27] Está diseñado para aplicaciones en tiempo real y tiene buenas funciones para imagen o video. Es muy recomendable para programadores que se centrarán en el campo de la visión por computadora, porque esta biblioteca puede crear aplicaciones que son confiables, sólidas en el campo de la visión digital y tienen capacidades similares al procesamiento visual en humanos. [28]

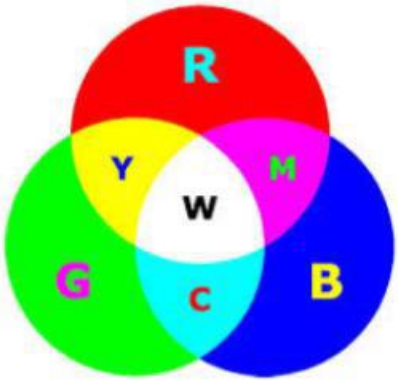
1.6.4. Detección de colores

Los "espacios de color" son la forma en la que se presentan los canales de color en la imagen, es decir, son quienes le dan a la imagen cada matiz particular. Existen diferentes espacios de color diferentes y cada uno tiene su propio significado, entre ellos están los espacios *RGB* (rojo, verde, azul), CMYK (cian, magenta, amarillo, negro) y *HSV* (tono, saturación, valor), etc.[19]

1.6.4.1. ***Espacio RGB***. El espacio de color predeterminado de *OpenCV* es RGB. Pero en realidad almacena color en formato *BGR*, es decir, se intercambia la posición de los canales azul y rojo. [29] Es un modelo de color aditivo, es decir la intensidad de cada canal determinara un color o tono diferente. Cada canal puede tomar un valor entre 0 y 255, donde el vector (0, 0, 0) representa el negro en su totalidad y el vector (255, 255, 255) representa el color blanco. [30]

Figura 1-2.

Espacio de color RGB[31].



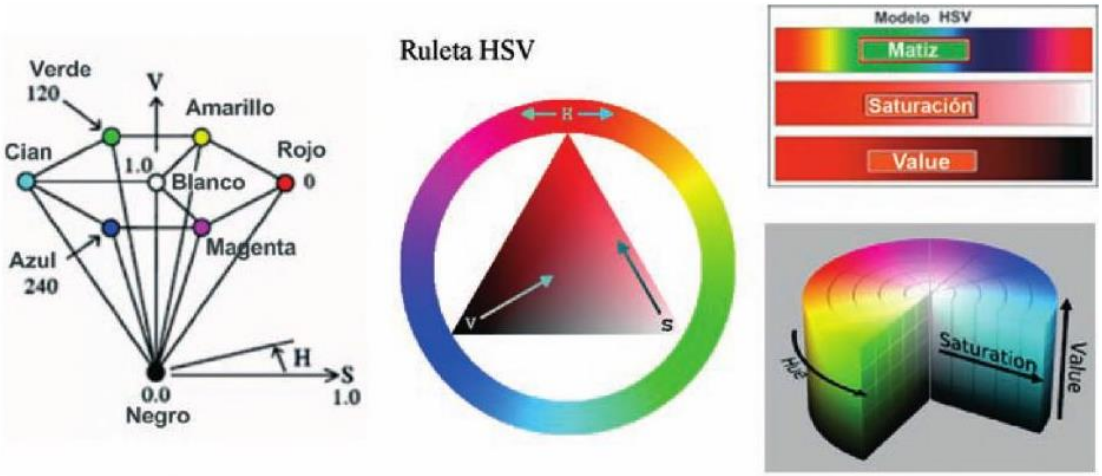
1.6.4.2.***Espacio HSV.*** El espacio de color HSV (*hue, saturation and value*; por sus siglas en inglés) es un modelo para representar el espacio de color similar al modelo de color *RGB*. [14]Pero sus canales tienen diferentes funcionalidades ya que el canal H o de matiz, modela el tipo de color, lo que es de gran utilidad en el procesamiento de imágenes para la segmentación de objetos en función de su color.

El canal S o de la saturación, se refiere a qué tan fuerte o qué tan presente es el color que vemos; pasa de no saturada a representar sombras de gris y completamente saturada.

El canal de valor V describe el brillo o la intensidad del color.[22]

Figura 1-3

Espacio de color HSV[32].



1.6.5. Detección de contornos

Para poder encontrar y dibujar contornos de algún objeto o región de interés, *OpenCV* emplea la función cv2.findcontours. La jerarquía de contornos demuestra la relación que tienen los contornos. [33] Se puede tener el caso en que exista un contorno y dentro de este se encuentre otro, y dentro otro. Entonces por ejemplo un contorno externo es padre, mientras que el interno es hijo.[30]

1.7. Librería PyQt

PyQt conecta el marco multiplataforma Qt C++ con el lenguaje Python, es un módulo *GUI*. Qt es más que un conjunto de herramientas *GUI*, por lo que presenta abstracciones de enchufes o subprocesos de red, junto con *unicode, SQL*, bases de datos, *SVG*, *OpenGL*, xml, un navegador web operativo, un sistema de servicio y una amplia gama de widgets *GUI*.[34] El principio sobre el que funciona una clase 'Qt' está relacionado con un mecanismo de ranura responsable de ofrecer comunicación entre elementos con el fin de diseñar componentes de software reutilizables con facilidad. Además, Qt viene con *Qt Designer*, una herramienta que actúa como una interfaz gráfica de usuario. [27]

PyQt puede diseñar código Python desde *Qt Designer*, mientras agrega nuevos controles GUI cuando se utilizan tanto *Qt Designer* como el lenguaje de programación Python. [24]

1.8. ISO/IEC 25010

Para que un software satisfaga los requerimientos de los usuarios aportando un valor, el mismo debe contar con un grado de calidad que incluya los siguientes requerimientos.[35]

1.8.1. Funcionalidad

Tiene completitud funcional lo que quiere decir que tiene un grado en el cual las funcionalidades cubren todos los objetivos propuestos, con corrección funcional por lo que el producto entrega los resultados deseados con la precisión deseada. Y pertinencia funcional ya que el producto realiza las tareas y funciones propuestas.[6]

1.8.2. Desempeño

Debe cumplir con comportamiento temporal, es decir que los tiempos de respuesta y procesamiento de un sistema tienen relación con un banco de pruebas establecido. Los recursos utilizados dentro del software con condiciones definidas cumplen su función. Además, la capacidad son los límites máximos de un parámetro cumplen con los requisitos propuestos.[24]

1.8.3. Compatibilidad

Dentro de este parámetro el producto puede convivir con otro software, compartiendo recursos sin daño alguno. Además de que dos o más sistemas pueden intercambiar información y utilizar la misma sin problemas.[35]

1.8.4. Usabilidad

Este requerimiento se refiere a la capacidad para reconocer su adecuación, donde el usuario puede entender si el software cumple con sus necesidades.[29] El usuario puede aprender cómo

aplicar, utilizar y controlar el producto con facilidad. Además, tiene protección contra errores de usuario con una interfaz es agradable y amigable con el usuario por lo que el producto puede ser utilizado por usuarios con discapacidades.[28]

1.8.5. Fiabilidad

El sistema es fiable en condiciones normales y accesible para su uso cuando se necesita. Además, el sistema funciona según lo previsto antes presencia de fallos de hardware o software, y puede recuperar los datos afectados y reestablecerse en caso de fallo.[25]

1.8.6. Seguridad

Posee protección ante accesos no autorizados y tiene la capacidad para prevenir accesos o modificaciones no autorizadas. Además, tiene la capacidad para mostrar las acciones o eventos realizados, para que puedan ser repudiados en el futuro. Sus capacidades incluyen el rastreo de acciones de una entidad y demostrar la identidad de un sujeto o un recurso.[24]

1.8.7. Mantenibilidad

Se trata de la capacidad para que ante un cambio en un componente haya un impacto mínimo en el conjunto y pueda ser utilizado en más de un sistema software. [35]Tiene facilidad para evaluar el impacto de un determinado cambio en el resto del software, diagnosticar las causas de fallos, o identificar las partes a modificar, por lo que puede ser modificado de forma eficiente sin problemas futuros. Puede determinar las pruebas a llevar a cabo y determinar si se cumplen los requerimientos.[25]

1.8.8. Portabilidad

El producto debe poder ser adaptado a diferentes entornos de hardware, software por lo que el producto se puede instalar y/o desinstalar de forma exitosa. De la misma manera puede ser utilizado en lugar de otro producto software determinado con el mismo propósito.[22]

CAPÍTULO II

2. MARCO METODOLÓGICO

En este capítulo se describe el proceso de programación y obtención del sistema para la detección de deficiencias nutricionales en el cultivo de mango, misma que está basada en el lenguaje *Python* debido a que es un lenguaje de programación de alto nivel el cual facilita el aprendizaje al igual que sus librerías de visión por computador.

2.1. Materiales y Métodos

En esta sección se evalúa los materiales utilizados en el trabajo de investigación, así como también los procedimientos, enfoques, diseños y tratamientos realizados en la investigación.

2.1.1. Hardware

El hardware utilizado en el presente trabajo de grado se detalla en la siguiente tabla. Ver tabla 2.

Tabla 2

Ordenador para programación

Modelo	Lenovo y520-15IKBN
Procesador	Intel Intel(r) Core (tm) i5-7300hq 2.50ghz
Memoria instalada (RAM)	8.00 GB.
Tipo de sistema	Sistema operativo de 64 bits Windows 10

2.1.2. Software

- Sistema operativo de 64 bits Windows 10
- Anaconda
- Spyder
- Python 3.7.3

2.2. Diseño, nivel y tipo de investigación

En esta sección se definen los detalles que caracterizan a esta investigación, en este caso será el diseño, nivel y tipo de investigación.

2.2.1. Diseño de la investigación

El método propuesto es experimentalmente descriptivo, porque es una pregunta que demuestra las características de un nuevo estudio, ofrece una mejor optimización de su tecnología, productos y materiales, nivel este nivel de investigación está guiado por la experimentación porque es el único método que hace es posible establecer relaciones de causa y efecto entre variables.[19]

2.2.2. Nivel de la investigación

El grado de investigación empírica se refiere a la profundidad de conocimiento que la investigación quiere adquirir, por lo que el nivel de esta investigación es exploratorio, lo que indica que la investigación exploratoria busca abrir nuevas vías en el desarrollo intelectual humano. Y la presente investigación, como diseño de un sistema, busca allanar el camino para un nuevo enfoque, una alternativa que consiste en diseñar un algoritmo de procesamiento de imágenes para brindar soluciones a la producción efectiva en el cultivo de mango.

2.2.3. Descripción de la investigación

Este estudio tiene como objetivo desarrollar o diseñar un algoritmo de procesamiento de imágenes para un sistema de identificación de deficiencias en el cultivo de mango, en este caso aplicado en la finca ‘Hernández’. Este estudio consta de cuatro pasos que se describen a continuación:

Etapa 1: esta etapa, se trata principalmente de investigación informativa, consulta de directorios y sitios web, fundamentación sobre el objeto a estudiar y los posibles algoritmos que se utilizarán en el proceso.

Etapa 2: incluye el inicio del desarrollo de algoritmos utilizando lenguaje Python y la librería OpenCV junto con 'numpy'.

Etapa 3: una vez tomados los algoritmos de procesamiento de imágenes, el propósito es realizar una interfaz amigable con el usuario, la cual muestre los resultados en pantalla de forma entendible.

Etapa 4: en esta etapa se realiza la validación de la investigación, por lo que es necesario recolectar muestras y así realizar pruebas de funcionamiento. Posteriormente se incluye el análisis de los resultados, seguido de la revisión del desempeño y la corrección posterior.

2.3. Recolección de datos.

La recolección de datos se refiere a las formas y medios utilizados para obtener información útil para corroborar las hipótesis.

Técnicas utilizadas:

- Revisión de directorios y bases de datos:

Esta técnica permitirá recolectar y sistematizar la información solicitada a; en internet se publican artículos, foros, blogs, libros, tesis, tutoriales en vídeo y otras fuentes de información.

- Observación:

Es importante utilizar esta técnica para recopilar información clave, ya que es posible interactuar con la realidad a través de observaciones como capturas de pantalla, proceso de diseño de algoritmos, notas de simulación y fotos.

2.4. Adquisición de las imágenes

Con el objetivo de diseñar el algoritmo de visión artificial, se obtienen imágenes de prueba, las cuales fueron obtenidas mediante un iPhone Xs Max con una resolución de 2337x3116 en formato '.jpg' en un ambiente controlado para no obtener distorsión o variación en el color a buscar. Para encontrar este ambiente controlado se utilizaron dos lámparas superiores y una lámina blanca como fondo. Se realiza la adquisición de 3 imágenes por cada deficiencia especifica.

2.5. Anaconda Navigator

Anaconda es una plataforma informática científica y de procesamiento de datos basada en Python. Ha incorporado muchas bibliotecas de terceros muy útiles. Instalar Anaconda es equivalente a instalar automáticamente Python y algunas bibliotecas de uso común, como Numpy, Pandas, Scrip y Matplotlib, por lo que hace que la instalación sea mucho más fácil que la instalación normal de Python.

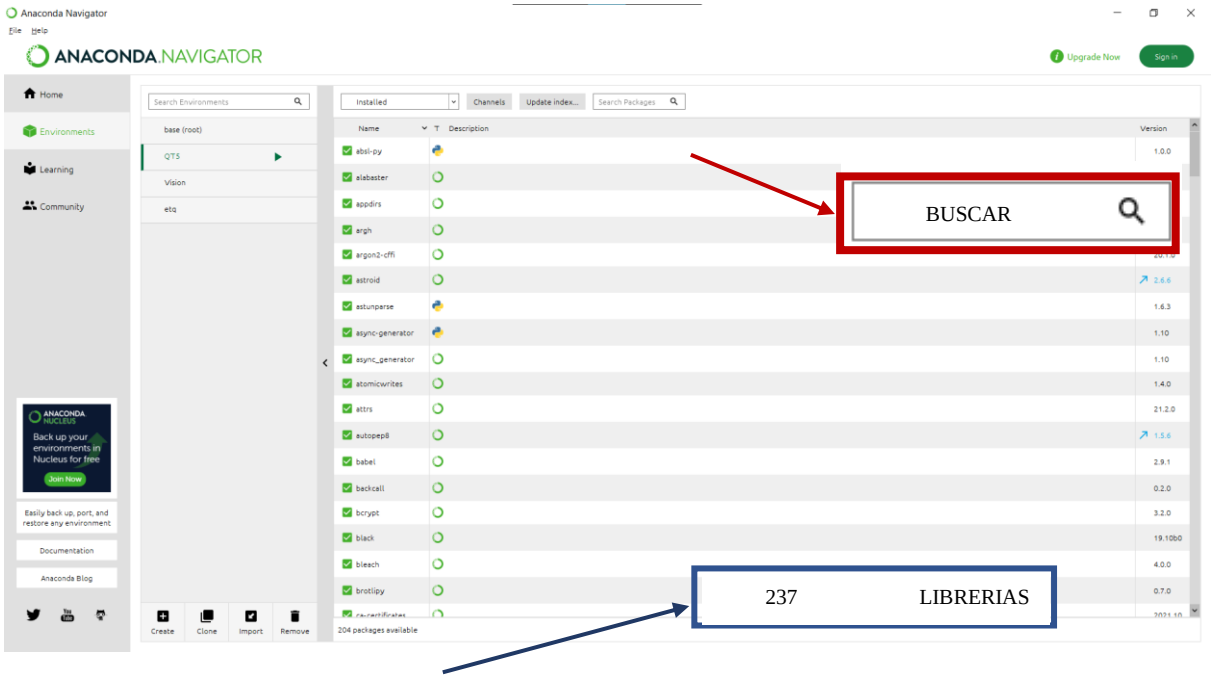
Es un software libre por lo que solo es necesario ingresar a la página oficial de Anaconda para descargar e instalar de acuerdo con el sistema operativo requerido.

Una vez instalado Anaconda se procede a crear un nuevo ambiente, sobre el cual se instalarán las respectivas librerías y software o aplicaciones que ayudan a realizar la programación.

En la figura 2-1 se puede visualizar la interfaz de Anaconda Navigator en la cual se puede buscar librerías (ver recuadro rojo de la fig.2-1) e instalarlas. En este caso se encuentran instalados 237 librerías (ver recuadro azul de la fig.2-1). Ver fig. 2-1.

Figura 2-1

Interfaz de Anaconda Navigator



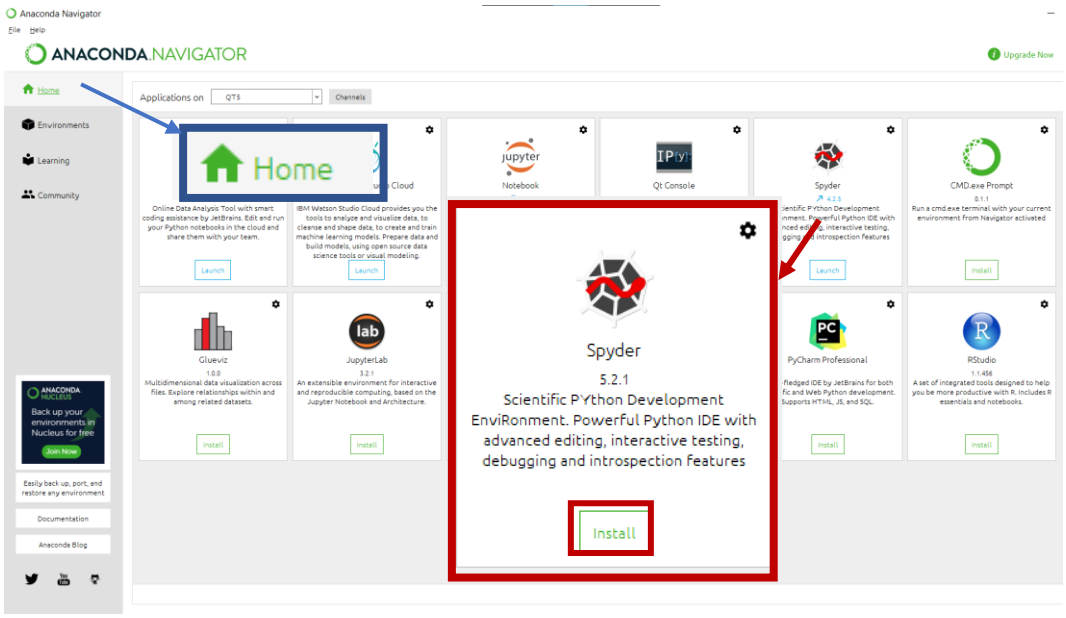
2.6. Spyder

Spyder es un IDE multiplataforma de código abierto. El cuál está escrito completamente en Python. Está diseñado por científicos y se trata de un software hecho exclusivamente para científicos, analistas de datos e ingenieros.

Para instalar Spyder en Anaconda se dirige a ***Home*** y clic sobre la aplicación en la palabra 'Install', Ver fig. 2-2. En este caso se instaló la versión Spyder 5.2.1. [45]

Figura 2-2

Instalación de Spyder.



2.7. Librerías

Las librerías de Python son diversos módulos las cuales permiten el acceso de funcionalidades específicas del sistema lo que brinda soluciones estandarizadas a problemas de programación. En este caso las librerías a instalar son numpy, OpenCV, y PyQt5 con sus respectivas herramientas. [24]

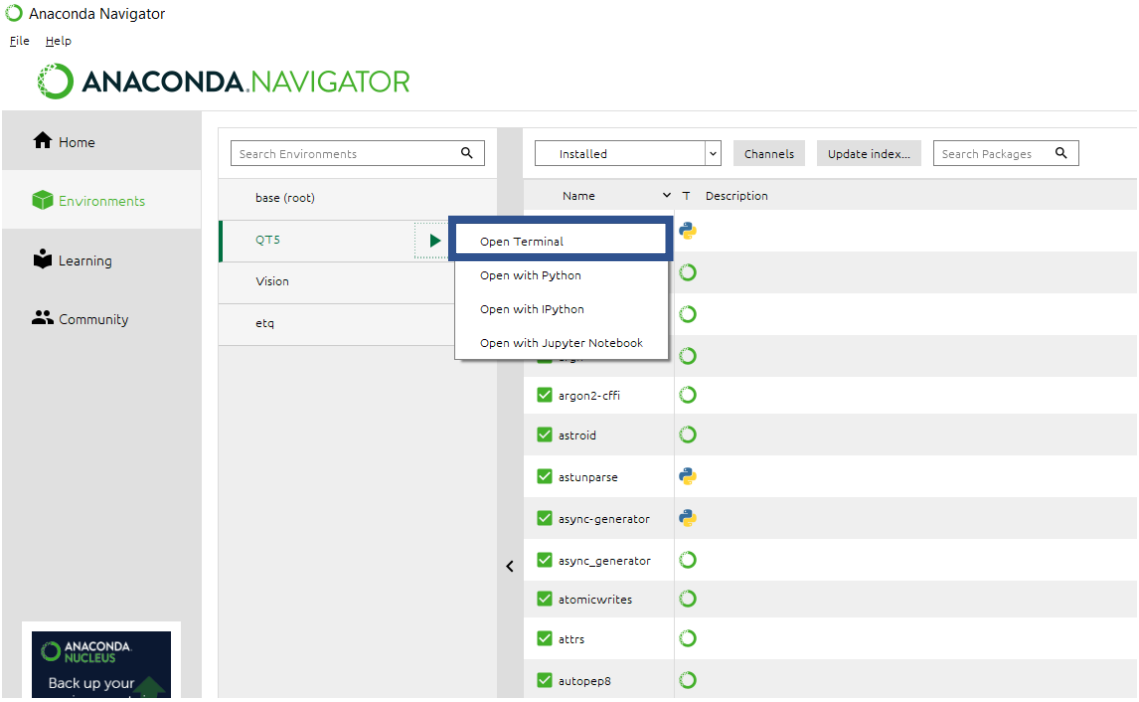
2.7.1 Numpy

Numpy es una librería para el lenguaje de programación que tiene como objetivo permitir la manipulación de vectores y matrices.

Para la instalación únicamente dentro del ambiente que se quiera instalar la librería, se abre la terminal como se muestra en la figura 2-3. Ver fig. 2-3.

Figura 2-3

Proceso de ingreso a terminal



Una vez dentro de la terminal únicamente se digita la siguiente línea de código.

```
pip install numpy
```

2.7.2. Open CV

Open CV proporciona una librería con distintas herramientas para procesamiento de imagen y video para la programación de visión por computadora optimizados en tiempo real.[29]

Para la instalación únicamente dentro del ambiente que se quiera instalar la librería, se abre el terminal como se mostró en la imagen. [24]

```
pip install opencv-python
```

2.7.3. PyQT5 Designer

Para su instalación se debe escribir en el terminal:

```
pip install pyqt5==5.12.3 ---user
```

2.8. Interpretación de las líneas de código

Una vez instaladas todas las librerías necesarias se procede a la escritura o diseño del algoritmo para la detección de deficiencias en el mango.

Las líneas de código se describen de la siguiente manera:

- Las líneas de código (8-11) del algoritmo importa las librerías OpenCV y numpy
- La línea de código 14 del algoritmo es para leer la imagen
- Las líneas de código (17-20) son para redimensionar la imagen de entrada y además se crea dos variables más con las mismas dimensiones.
- La línea de código 22 se realiza tratamientos a la imagen para de esta manera hacer los colores más uniformes. El tratamiento es un Blur Gaussiano. Ver Figura 2-4.

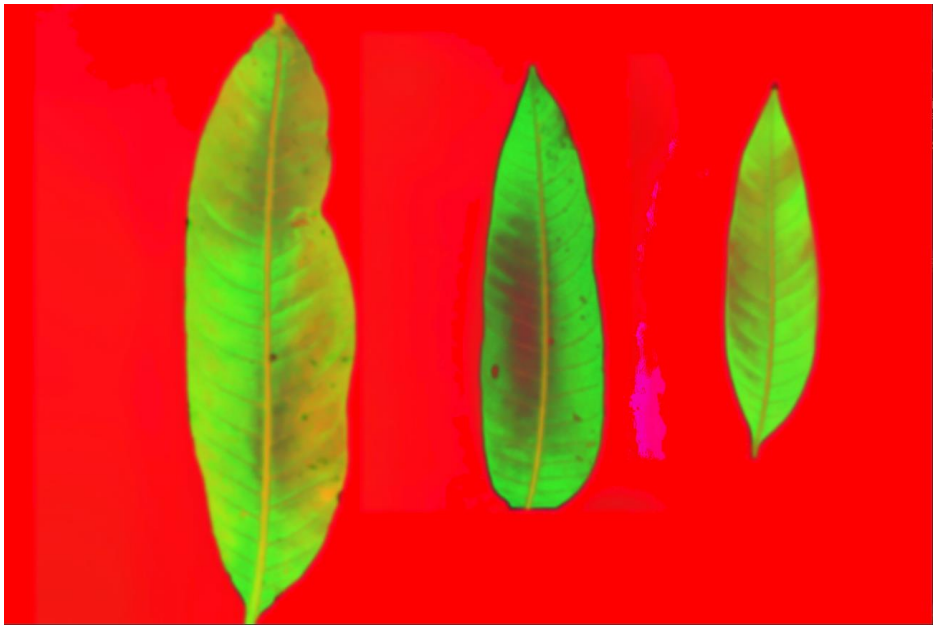
Figura 2-4.

Tratamiento *Blur* aplicado a muestra.

- La línea 26 del código convierte el espacio de color de la imagen de entrada de *RGB* a *HSV*. Ver fig. 2-5.

Figura 2-5

Conversión a espacio de color HSV de RGB.



- La línea de código 29 del algoritmo obtiene el plano de saturación a partir de la imagen *HSV*. Ver fig. 2-6.

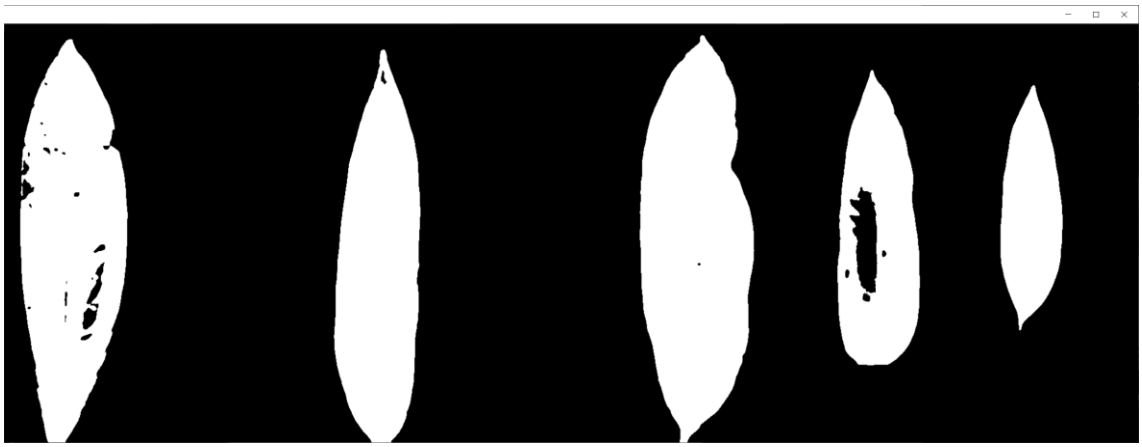
Figura 2-6

Obtención del plano de saturación a partir de HSV.

- La línea de código 32 del algoritmo aplica un umbral en plano de saturación. Se muestra su resultado en la figura 2-7. Ver fig. 2-7.

Figura 2-7

Aplicación de umbral.



- La línea de código 35 del algoritmo crea una variable contornos, con el algoritmo cv2.findContours a partir del umbral de saturación.
- La línea de código 38 del algoritmo dibuja los contornos encontrados de color negro y grosor 10. Ver fig. 2-8.

Figura 2-8

Contornos obtenidos de cada muestra.

- La línea de código 40 inicializa la variable ‘i’ que servirá como un contador más adelante.

- La línea de código (41-147) del algoritmo se trata de un algoritmo de iteración en cada contorno encontrado. En este caso cada contorno es una hoja de muestra.
- Las líneas de código 42-49 del algoritmo sirven para encontrar el área del contorno que se está tratando con sus respectivas coordenadas para posteriormente realizar el recorte de la imagen.
- Las líneas de código 51-58 del algoritmo sirven para definir las variables de colores, en este caso los rangos de colores en los que se encuentra cada deficiencia, haciendo uso de la librería de numpy.
- La línea de código 60 del algoritmo sirve para convertir el formato de color de imagen desde RGB a HSV
- Las líneas de código (62-64) del algoritmo sirven para obtener imágenes con las máscaras de acuerdo con el rango de color antes expuesto a partir de la imagen en HSV.
- La línea de código 66 sirve para encontrar el contorno de acuerdo con la máscara obtenida en el rango de color de la deficiencia de nitrógeno.
- Las líneas de código (68-85) del algoritmo sirve para trabajar en el contorno de nitrógeno.
- La línea de código 70 obtiene el área total del contorno de nitrógeno que se esté procesando.
- Las líneas de código (71-78) son un condicional en el cual si el área trabajada es mayor al 70 por ciento del área total de la hoja se procede a obtener el centroide de su contorno
- Las líneas de código (79-85) son un nuevo condicional en el cual se le indica al programa que, si el centroide anteriormente encontrado se encuentra en el área central del contorno de la hoja, realice un contorno de esta área y se muestre en consola la palabra ‘Nitrógeno’

- La línea de código 87 sirve para encontrar el contorno de acuerdo con la máscara obtenida en el rango de color de la deficiencia de potasio.
- Las líneas de código (90-111) del algoritmo sirve para trabajar en el contorno de nitrógeno.
- La línea de código 90 obtiene el área total del contorno de nitrógeno que se esté procesando.
- Las líneas de código (94-111) son un condicional en el cual si el área trabajada es mayor a la quinta parte del área total de la hoja se procede a obtener el centroide de su contorno
- Las líneas de código (98-111) son un nuevo condicional en el cual se le indica al programa que, si el centroide anteriormente encontrado se encuentra en el área superior del contorno de la hoja, realice un contorno de esta área y se muestre en consola la palabra ‘Fósforo’
- La línea de código 114 sirve para encontrar el contorno de acuerdo con la máscara obtenida en el rango de color de la deficiencia de potasio.
- Las líneas de código (117-136) del algoritmo sirve para trabajar en el contorno de potasio encontrado previamente.
- La línea de código 118 obtiene el área total del contorno de nitrógeno que se esté procesando.
- Las líneas de código (119-136) son un condicional en el cual si el área trabajada es mayor a la quinta parte del área total de la hoja se procede a obtener el centroide de su contorno
- Las líneas de código (124-136) son un nuevo condicional en el cual se le indica al programa que, si el centroide anteriormente encontrado se encuentra en el área central

del contorno de la hoja, realice un contorno de esta área y se muestre en consola la palabra ‘Potasio’

- Las líneas de código 138 del algoritmo inicializa un string compuesto de la palabra hoja, el numero de la variable ‘i’ más la extensión de la imagen
- La línea de código 139 guarda la imagen con el nombre anteriormente compuesto y el contorno de la hoja que se esté trabajando
- Las líneas de código 143 y 144 obtiene las coordenadas del centroide del contorno de la hoja que se esté trabajando
- La línea de código 145 imprime en consola el número de la hoja que se esté trabajando
- La línea de código 146 sirve para poner texto en la imagen a mostrar, en este caso, escribe la palabra ‘Hoja’ y el correspondiente número.
- La línea de código 147 del algoritmo sirve como contador, el cual va aumentando en 1 por cada contorno encontrado.

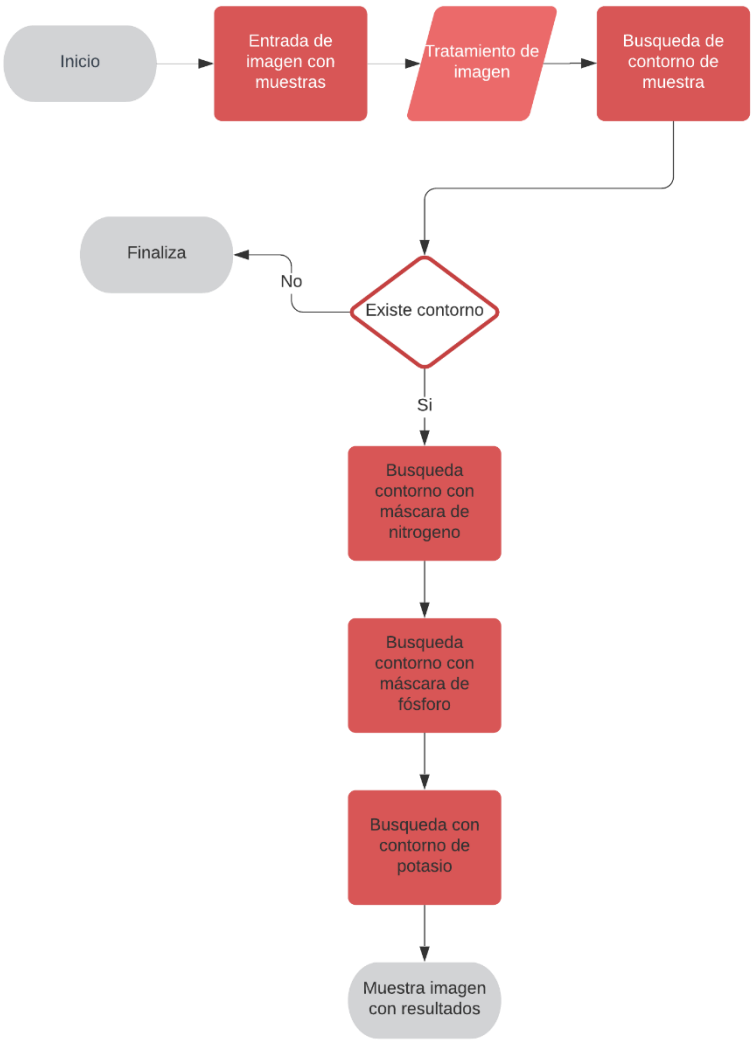
Las líneas de código 148-151 sirven para mostrar los resultados obtenidos, en este caso son la imagen final con la detección de deficiencias y las máscaras de cada deficiencia.

2.9. Diagrama de flujo del algoritmo

A continuación, se presenta el diagrama de flujo del algoritmo principal que se implementará en la interfaz, el cual describe el proceso de ejecución y funcionamiento del sistema realizado en la herramienta de diagramación en la web *Lucidchart*. Ver fig. 2-9.

Figura 2-9

Diagrama de flujo de algoritmo principal.



2.10. Interfaz de usuario

La interfaz de usuario debe ser intuitiva y fácil de usar por lo que debe tener la menor cantidad de texto posible de tal manera que los botones y etiquetas a utilizar sean gráficos en su mayoría.

2.11. QT Designer

Una vez instalada la versión más reciente de PyQt5, y el editor gráfico Qt Designer.

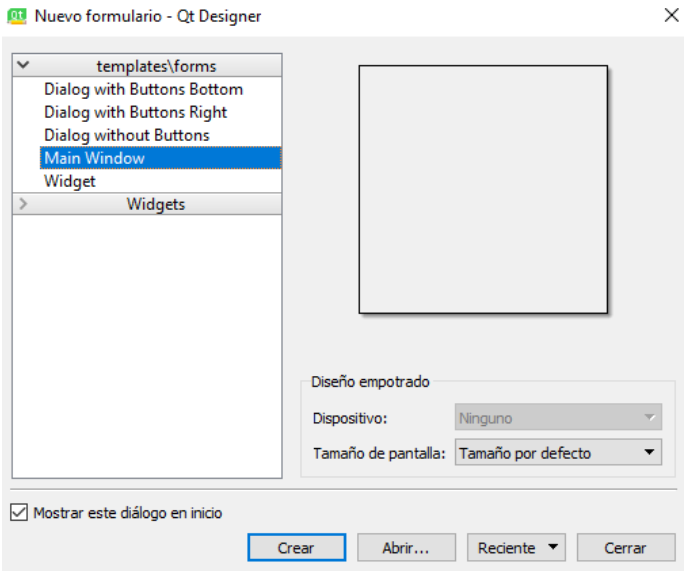
Para abrirlo se debe escribir en el terminal:

```
designer
```

Se abre el programa Designer y en primera instancia da a elegir qué tipo de ventana se va a diseñar. Ver fig. 2-10. Para este caso, se selecciona "*Main Window*" la cual representara la raíz del programa gráfico.

Figura 2-10.

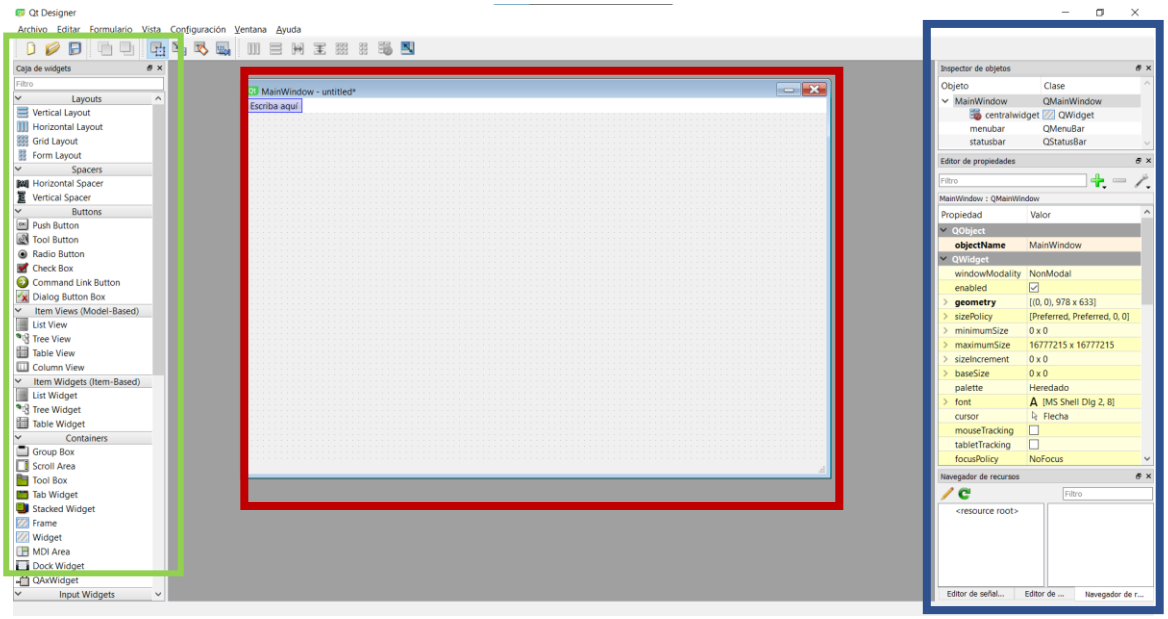
Nuevo formulario *Main Window* en *Designer*.



Al presionar crear, la interfaz se divide en cuatro áreas, ver figura 2-11. El área de control, de color verde, donde se encuentran todos los widgets que se pueden utilizar para el diseño del sistema. El área de proyecto, de color rojo, donde se arrastran los elementos deseados y área de propiedades, de color azul, donde se puede inspeccionar cada objeto añadido a la interfaz de tal manera que se pueda editar sus atributos.

Figura 2-11

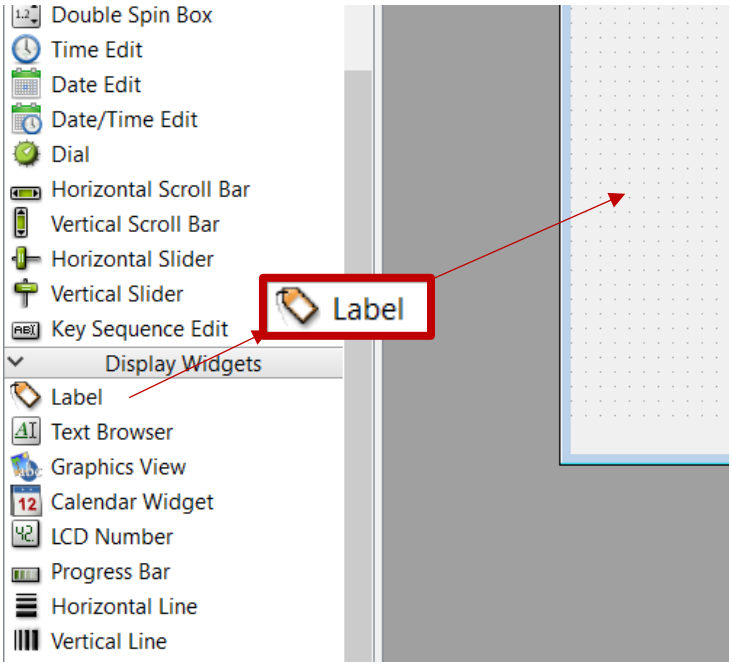
Áreas de trabajo en Designer.



A partir del área de control se toman los elementos necesarios, en este caso, se toman las etiquetas. Estas etiquetas servirán para mostrar títulos de los elementos. Ver fig. 2-12.

Figura 2-12

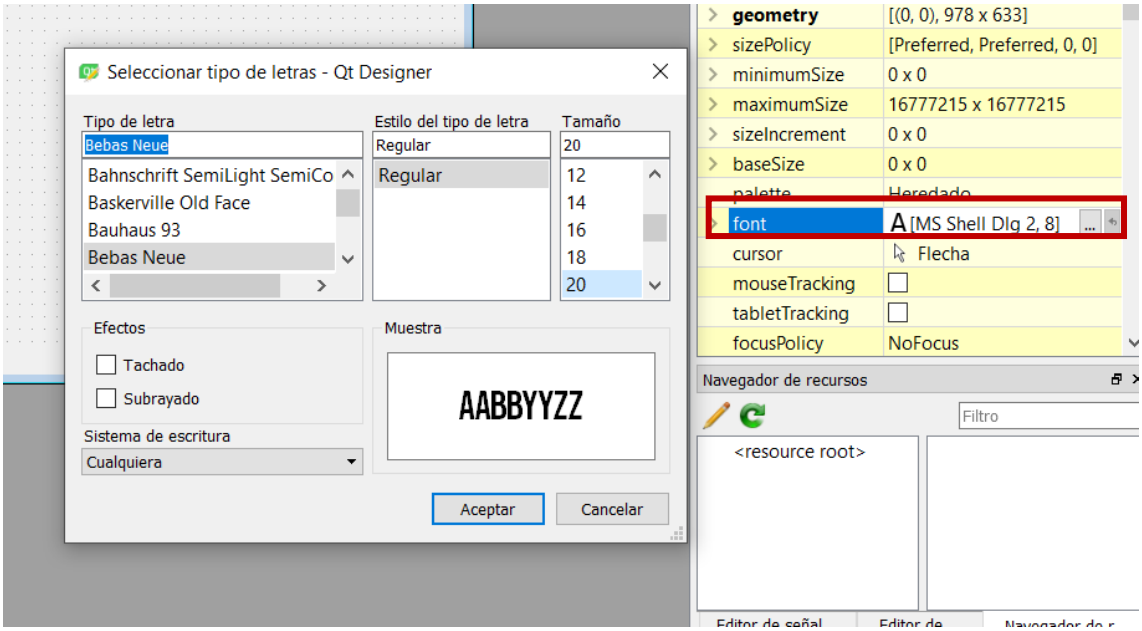
Selección de elemento *label* en *Designer*.



A continuación, en el área de propiedades se realizan los cambios respectivos del nombre de elemento y se selecciona la fuente y tamaño del texto. En este caso se ha seleccionado un tamaño considerable ya que se trata del título de la aplicación y como fuente se ha seleccionado ‘Bebas Nueu Bold’. Ver figura 2-13.

Figura 2-13

Selección de propiedades para '*label*' de texto.

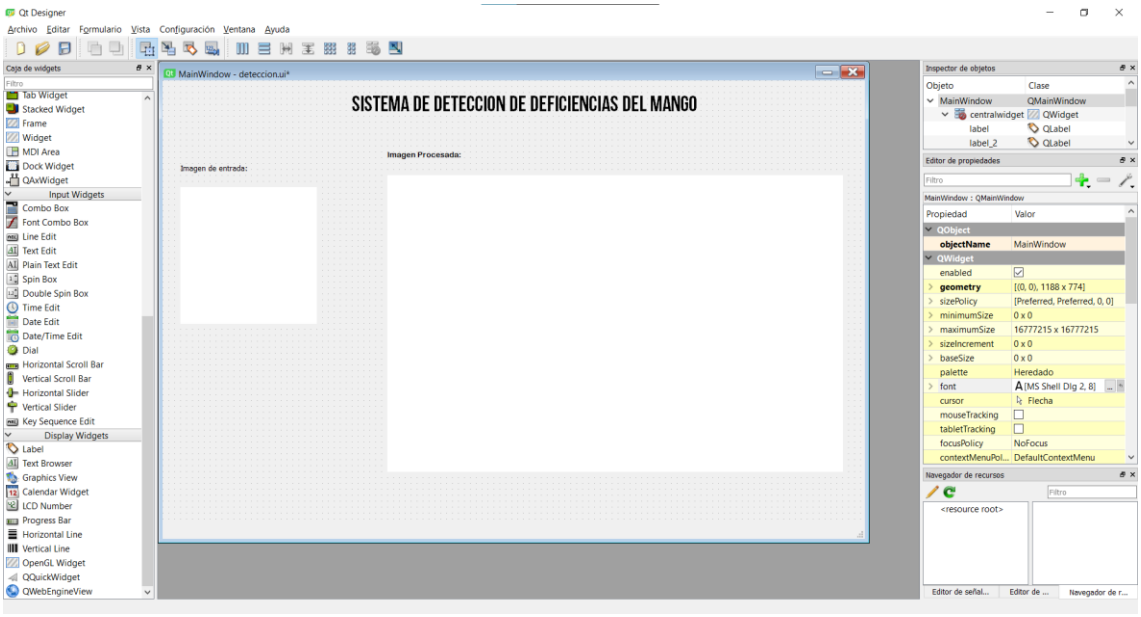


Esto se realiza con cada *label* de texto necesario para nuestro diseño, siempre considerando el tamaño necesario para que sea funcional.

Siguiendo con el diseño del sistema, son necesarios labels donde se mostrarán la imagen a procesar e imagen procesada.

Figura 2-14

Main Window con todos los elementos gráficos y textuales.

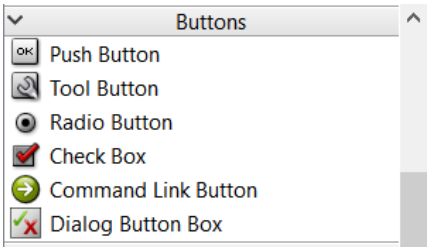


Se ha implementado en el diseño labels de imagen de tamaños diferentes y con la ayuda de los labels de texto es posible entender el propósito de cada uno. Ver fig. 2-14.

Por último, es necesario añadir botones que cumplan con funciones específicas cada uno para ello en el área de control se encuentran todos los botones disponibles. Ver fig. 15.

Figura 2-15

Inserción de botones en *Main Window*.



En este caso se seleccionó el *Push Button* ya que es el principal elemento para hacer que el sistema funcione correctamente. Son necesarios como botones principales, el botón de añadir imagen y botones respectivos para el procesamiento o detección de cada una de las deficiencias.

2.11.1. Exportando diseño de GUI a .py

Una vez finalizado el diseño es necesario guardarlo en un formato '.ui' para posteriormente realizar la transformación hacia un archivo .py, de tal manera que sea posible editarlo desde el código principal.

Para realizar la conversion de .ui a .py se utiliza el terminal en el cual se encuentra instalado la librería de PyQt5. Una vez en el terminal se introduce la siguiente línea de código.

```
pyuic5 'nombre_archivo_ui.ui' -o 'nombre_archivo.py'
```

donde:

- pyuic5: versión *Pyqt*, el número cambiará según la versión instalada.
- nombre_archivo.ui: Nombre del archivo creado en *QT Designer*.
- Nombre_archivo.py: Nombre del archivo que se creara una vez convertido.

En este archivo .py se realiza la programación de los botones y labels que se encuentran en el diseño.

Como primer punto se realiza la programación del botón para escoger la imagen a utilizar, la programación de este botón consta de 10 líneas de código:

- La primera línea inicializa una variable global llamada 'filename', en la cual se almacenará la dirección de la imagen a utilizar
- La segunda línea define la variable antes mencionada con la ayuda de las librerías de PyQt, añadiendo un filtro de Imagen.
- La tercera línea indica un condicional en el que si existe una ruta de archivo se puede continuar con las siguientes líneas, caso contrario, no realizará ninguna acción.

- En la cuarta línea de código se inicializa una variable ‘self.captured’ de manera que se pueda utilizar en los botones posteriores y con la librería de OpenCV se lee la ruta del archivo.
- En la siguiente línea se realiza una conversión de colores de la imagen anterior de BGR a RGB
- En la sexta línea se obtiene variables de filas, columnas y canales a partir de la forma de ‘self.captured’.
- En la séptima línea se inicializa los bytes por línea multiplicando el valor de filas por columnas
- En la octava línea se inicia una nueva variable y usando la librería QImage, esta nueva variable tiene como parámetros los datos de la imagen en RGB, los datos de columnas, filas, bytes, y le da un formato RGB.
- En la novena línea se escala la imagen a mostrar de acuerdo a la dimensión en donde se va a mostrar, en este caso, obtiene las dimensiones del label y mantiene el aspecto de radio.
- En la décima línea, con las librerías QyGui y QPixmap se coloca la imagen en el label escogido.

El siguiente botón para programar es el de nitrógeno, en el cual se utiliza el algoritmo desarrollado en el prefacio 2.8, pero únicamente se mantiene la secuencia y variables correspondientes a la deficiencia de nitrógeno.

Adicionalmente se debe añadir líneas de código para que una vez se procese la imagen sea posible mostrarla en el label y guardar esta imagen para posteriormente poder visualizar con el software predeterminado del sistema operativo.

Para la programación de los botones de fósforo y potasio se realiza el mismo proceso anterior en el cual solo se mantiene la secuencia y variables correspondientes a la deficiencia que se esté programando, conjuntamente se debe añadir las líneas de código para mostrar y guardar la imagen procesada.

Al final de cada botón es necesario definir la ruta de la imagen que se está mostrando en el label de la imagen procesada para poder utilizarla en el botón de visualización.

CAPÍTULO III

3. PRUEBAS Y ANÁLISIS DE RESULTADOS

En este capítulo se presenta los resultados y pruebas realizadas para verificar el correcto funcionamiento del algoritmo y de todos los elementos de la interfaz gráfica de usuario.

3.1. Resultados

En consecuencia, una vez culminada la etapa de desarrollo de la interfaz gráfica de usuario, se obtuvo como resultado una propuesta de interfaz para la detección de deficiencias nutricionales en el cultivo de mango, a continuación, se describe el contenido de la GUI desarrollada.

3.2. Diseño de la interfaz gráfica de usuario

En relación con el diseño realizado de la interfaz gráfica de usuario, siguiendo con lo mencionado en el epígrafe que hace mención a la herramienta de PyQt. Se obtiene una única ventana con distintas funcionalidades para cada botón, cada uno funcionando correctamente.

Se presenta la ventana resultante del diseño realizado en QT Designer y programación de elementos de la interfaz mediante Spyder. Ver fig. 3-1.

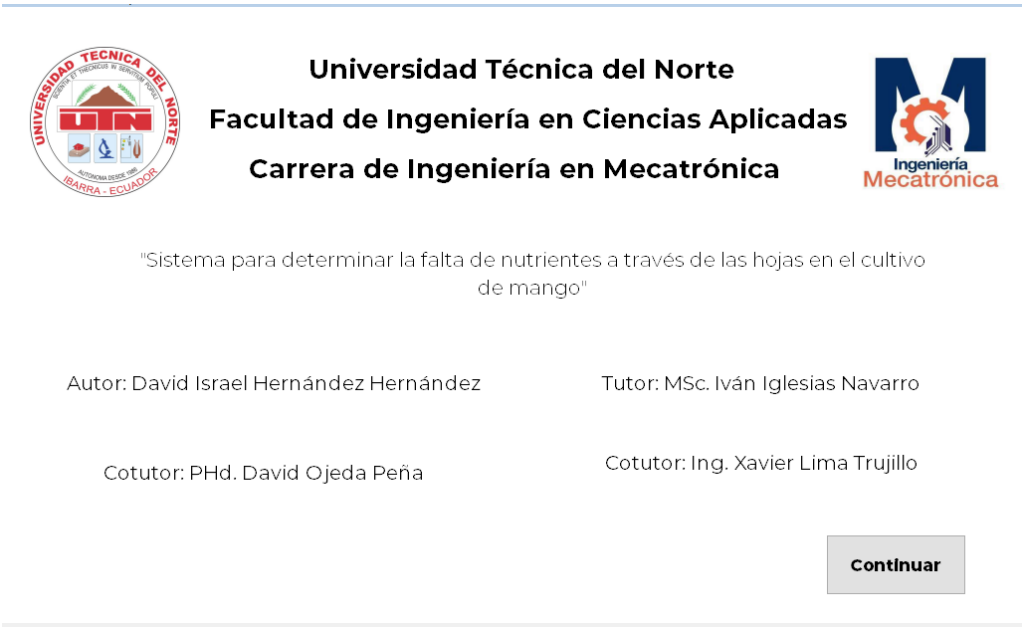
Figura 3-1

Main Window con todos los elementos funcionales.

Se muestra la interfaz gráfica de usuario como una ventana de introducción al usuario para que sea posible la identificación del software a utilizar. Por lo que se indica información visual de la procedencia del trabajo, en este caso, se visualiza en logos oficiales la institución universitaria, facultad y carrera universitaria. Por otro lado, de forma escrita en etiquetas o *labels* se presenta el título del trabajo de investigación de donde procede la interfaz gráfica de usuario. De la misma manera son presentados el autor, tutor y cotutores del trabajo de investigación. Finalmente, en la parte inferior izquierda se visualiza el botón funcional para continuar y realizar los análisis que se requieran. Ver fig. 3-2.

Figura 3-2

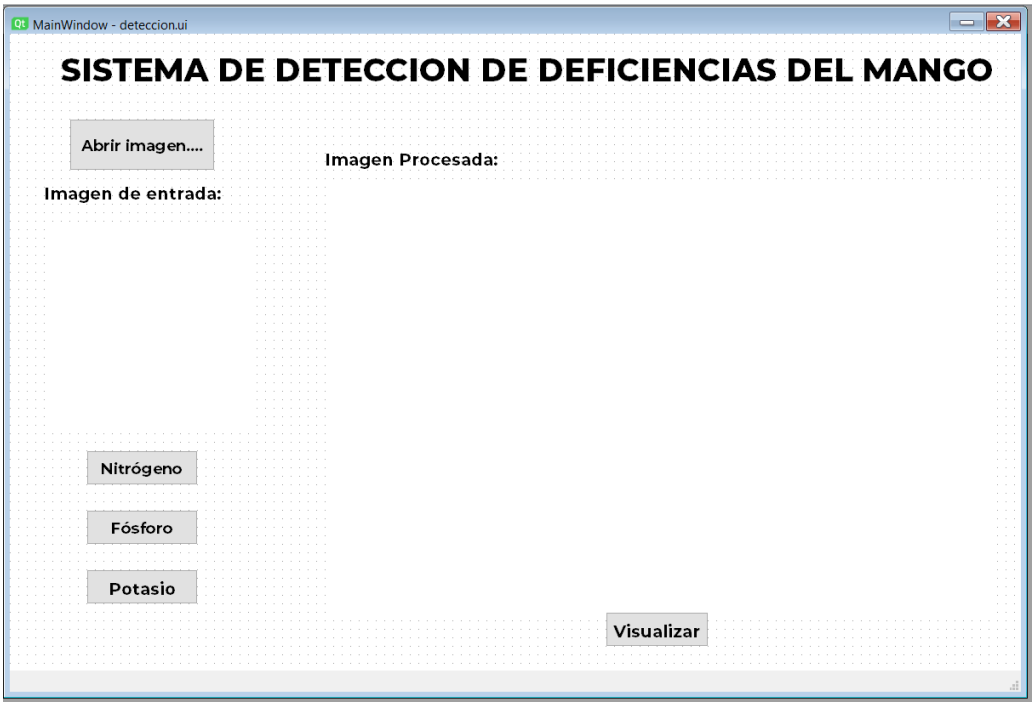
Portada de sistema



Se muestra la interfaz gráfica que realiza el cálculo de deficiencias nutricionales del mango a través de sus hojas. Ver fig. 3-3.

Figura 3-3

Interfaz principal del sistema.



A primera vista se puede apreciar los labels para título e imágenes a mostrar. Los labels de color blanco son aquellos donde se muestra la imagen a procesar y la imagen procesada. Se cuenta también con 4 botones, cada uno con una función específica, empezando por el botón 'Abrir imagen' el cual nos muestra un navegador de archivos en el cual podremos seleccionar la muestra o muestras dependiendo del requerimiento del usuario, una vez pulsado el botón se mostrará, en el *label* blanco y de menor tamaño, la imagen seleccionada. Después se tiene los botones para detectar cada una de las deficiencias, una vez pulsado cualquiera de estos botones podremos observar si alguna de estas muestras tiene deficiencia o no.

Para la visualización se tiene un botón, el cual muestra la imagen con el programa predeterminado de visualización de imágenes del sistema operativo, de tal manera que el usuario sea capaz de visualizar de mejor manera la imagen procesada.

3.3. Pruebas

Con el diseño y desarrollo de la interfaz gráfica de usuario culminada, se realizan las pruebas de funcionamiento para comprobar que cada botón de procesamiento y etiqueta realice la acción para la cual fue programada como se indica a continuación.

Figura 3-4

Diálogo del sistema para abrir imagen.

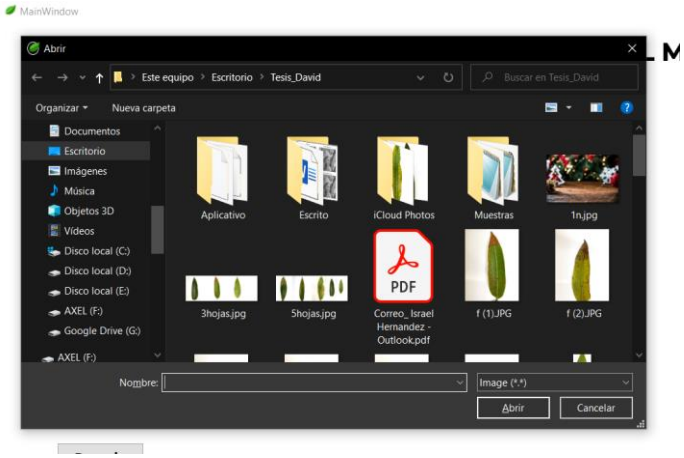


Figura 3-5

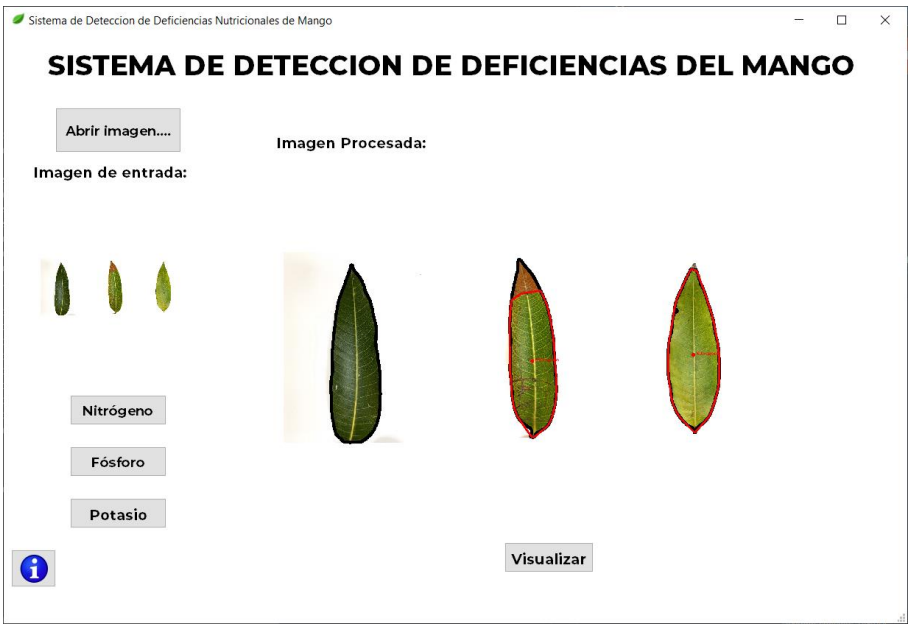
Imagen de muestra situada en *label* de entrada.

Como se puede ver en las figuras 3-4 y 3-5 se tiene que el botón de abrir imagen cumple su función correctamente al elegir la imagen y mostrarla en el *label* que está programado.

Seguidamente, en la figura 3-6 se puede visualizar el funcionamiento del botón de nitrógeno, el cual muestra la imagen procesada con su respectiva detección. Para su prueba se ha tomado una fotografía con 5 hojas como muestra. Ver fig. 3-6.

Figura 3-6

Funcionamiento botón: Nitrógeno.



A continuación, se puede visualizar el funcionamiento del botón de fósforo, el cual muestra la imagen procesada con su respectiva detección. Para la presente prueba se realiza una captura fotográfica con 3 hojas como muestra. Ver fig. 3-7.

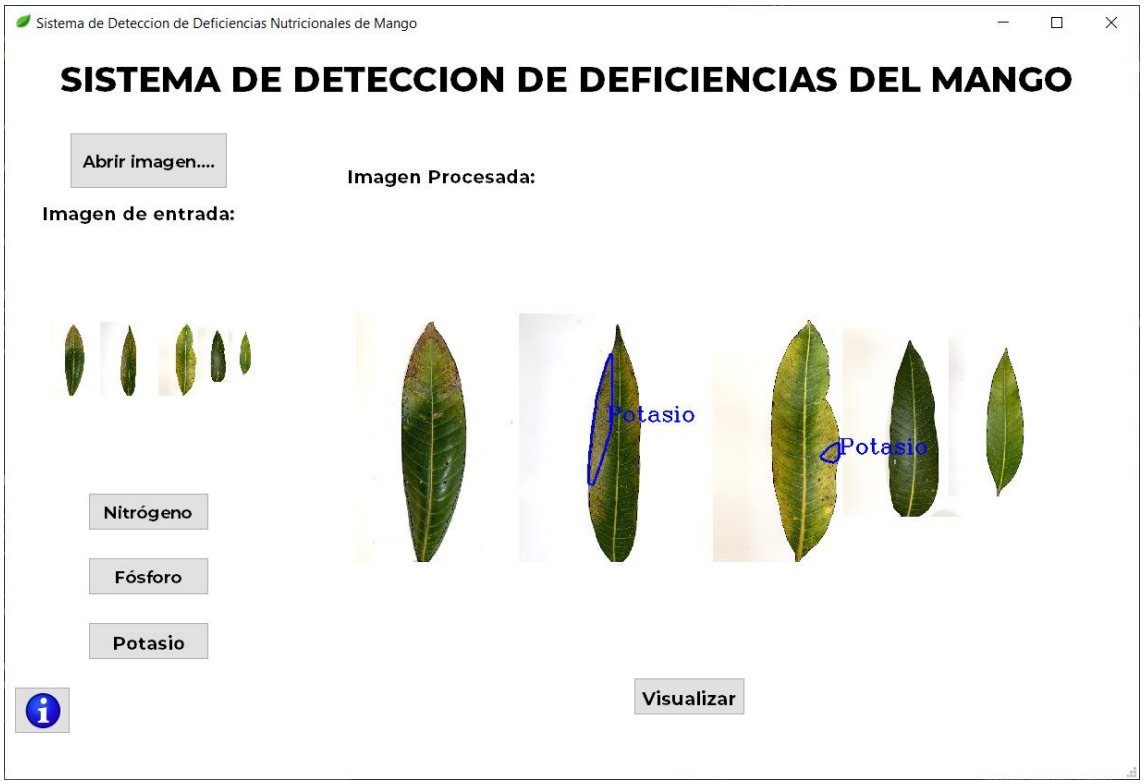
Figura 3-7

Funcionamiento botón: Fósforo

En lo que respecta a procesamiento se tiene la prueba del último botón, que respecta al botón de potasio. Ver fig. 3-8. se visualiza su funcionamiento, el cual muestra la imagen procesada con su respectiva detección. Para su prueba se ha tomado una fotografía con 5 hojas como muestra.

Figura 3-8

Funcionamiento botón: Potasio.



Finalmente, se tiene el botón ‘Visualizar’. Ver fig. 3-9. Cumple con su funcionamiento.

Figura 3-9

Funcionamiento botón: Visualizar

CAPÍTULO IV

4.1.Validación del algoritmo

Para verificar el desempeño de clasificación se hace uso de matrices de confusión las cuales están conformadas por una proporción de verdaderos positivos, con el cual se puede medir la efectividad del clasificador, y la proporción de verdaderos negativos, que muestra la especificidad del clasificador.

4.2.Validación algoritmo deficiencia de nitrógeno

Para la realización de esta matriz de confusión se utilizaron 88 muestras de hojas con distintas deficiencias, de entre ellas, la deficiencia de nitrógeno.

Es posible visualizar los resultados de esta matriz una vez procesadas todas las muestras. Ver tabla 3.

$$\frac{Verdaderos\ positivos}{(Verdaderos\ positivos+Falsos\ negativos)} \text{ (Ecuación 1)}$$

$$\frac{Verdaderos\ negativos}{(Verdaderos\ negativos+Falsos\ positivos)} \text{ (Ecuación 2)}$$

Tabla 3

Matriz de confusión- Algoritmo de Nitrógeno

	Nitrógeno	
	Si	**No**
Si	23	2
No	8	55
Total muestras		88

Para conocer su conocer su eficacia es necesaria la división del número de verdaderos positivos entra la suma de verdaderos positivos y falsos negativos.

Por otro lado, para conocer su especificidad es necesaria la división del número de verdaderos negativos entre la suma de verdaderos negativos y falsos positivos.

4.3.Validación algoritmo deficiencia de fósforo

Se muestran los resultados de la matriz de confusión para el algoritmo de fósforo una vez todas las muestras fueron procesadas. Para la realización de esta matriz de confusión se utilizaron 88 muestras de hojas con distintas deficiencias, de entre ellas, la deficiencia de fósforo. Ver tabla 3.

Tabla 4

Matriz de confusión- Algoritmo de Fósforo

	Fósforo	
	Si	**No**
Si	6	10
No	11	61
Total muestras		88

De igual manera es necesario conocer su eficacia y especificidad por lo que se utiliza la ecuación 1 y ecuación 2, respectivamente para cada valor.

4.4. Validación algoritmo deficiencia de potasio

Se muestran los resultados de la matriz de confusión para el algoritmo de potasio una vez todas las muestras fueron procesadas. Para la realización de esta matriz de confusión se utilizaron 88 muestras de hojas con distintas deficiencias, de entre ellas, la deficiencia de potasio. Ver tabla 4.

Tabla 5

Matriz de confusión- Algoritmo de Potasio

	Potasio	
	Si	**No**
Si	14	15
No	5	54
Total muestras		88

4.5. Análisis de resultados

Una vez realizadas las correcciones de errores en el algoritmo de procesamiento y la interfaz gráfica y por medio de las matrices de confusión se obtienen los resultados de eficacia y especificidad.

4.5.1. Resultados algoritmo de nitrógeno

A partir de la matriz de confusión de nitrógeno se tiene como resultado una eficacia del 74,2% y una especificidad del 96,5%.

El resultado de eficacia se debe a que el valor de falsos negativos obtenidos durante las pruebas es mayor a la tercera parte del número de verdaderos positivos.

El algoritmo de nitrógeno posee ciertas restricciones de área para su contorno por lo que debe

cumplir esa condición para que sea considerada como una deficiencia. Esto produce que al tener un conjunto de más deficiencias en la misma hoja que ocupan un área considerablemente grande de la hoja no se pueda considerarla como tal.

Ver fig. 4-1. Se puede visualizar como el algoritmo de detección de deficiencia de nitrógeno falla debido a la presencia de otra deficiencia.

Figura 4-1.

Muestra con deficiencias múltiples

A continuación, se visualiza una muestra, la cual se encuentra en la misma situación que la anterior figura; pero en este caso la segunda deficiencia ocupa un área menor al total de la hoja por lo que el algoritmo de detección de deficiencia de nitrógeno será capaz de detectarla como tal. Ver fig. 4-2.

Figura 4-2

Muestra con deficiencias múltiples

Al ser una detección basada en colores, se requiere de un ambiente controlado para eliminar cualquier tipo de interferencia en la luz por lo que cualquier variación en el ambiente provocaría que los colores cambien y la detección sea menos eficiente.

Ver fig. 4-3. En este caso se tiene una imagen la cual tiene una posible deficiencia de nitrógeno debido a su color, pero por condiciones ambientales se produce sombra sobre la hoja, lo que produce un cambio en su color, por lo tanto, el algoritmo no la detectará como una deficiencia de nitrógeno.

Figura 4-3

Muestra con deficiencia de nitrógeno no detectable.

El algoritmo funciona correctamente ya que la fotografía tiene un ambiente controlado y sus coloraciones no han sido alteradas. Ver fig. 4-4.

Figura 4-4

Muestra con deficiencia de nitrógeno detectable.

4.5.2. Resultados algoritmo de fósforo.

A partir de la matriz de confusión de fósforo se obtienen los resultados de 73,7% para eficacia y 78,3% de especificidad.

El resultado de eficacia se debe a que el valor de falsos negativos obtenidos durante las pruebas es mayor a la tercera parte del número de verdaderos positivos. De igual manera el resultado de especificidad refleja que el valor de falsos positivos es considerable con respecto al valor de verdaderos negativos.

El algoritmo posee cierto tratamiento de imágenes el cual sirve para la detección de bordes y contornos, eliminando la mayor cantidad de interferencias posibles con el fin de dejar únicamente la hoja ya que es el principal objeto de estudio. Este tratamiento produce que algunos colores se pierdan como es el caso del color plomo.

Como se menciona en el prefacio que hace referencia a las consecuencias de la deficiencia de fosforo, esta produce necrosis en el ápice de la hoja lo que visualmente se traduce en la muerte de la hoja, y si esta es muy avanzada, se tornará de color plomo.

Ver fig. 4-5. La deficiencia de fósforo es perceptible para el ojo humano, pero debido al tratamiento de imagen realizado, se tiene como respuesta del algoritmo un falso negativo, esto ya que el área ocupada por la deficiencia tiene el color plomo casi en su totalidad.

Figura 4-5

Muestra con deficiencia de fósforo con falso negativo

A diferencia de la figura 4-5, en la que se muestra también de la misma manera la necrosis en el ápice de la hoja, pero también se puede notar que no es un color uniforme y presenta variaciones las cuales si son detectables para el algoritmo. Ver fig. 4-6.

De la misma manera, se requiere de un ambiente controlado para eliminar cualquier tipo de interferencia en la luz por lo que cualquier variación en el ambiente provocaría que los colores cambien y la detección sea menos eficiente.

Aunque generalmente al tratarse de una necrosis, los colores tienden a ser bastante diferenciables de las demás partes de la hoja por lo que los falsos negativos son mayormente producidos por el primer caso.

Figura 4-6

Muestra con deficiencia de fósforo de verdadero positivo

4.5.3. Resultados algoritmo de potasio

A partir de la matriz de confusión de potasio se obtienen los resultados de 64,1% para eficacia y 85,9% de especificidad.

El resultado de eficacia se debe a que el valor de falsos negativos obtenidos durante las pruebas es mayor a la mitad del número de verdaderos positivos.

Este resultado nuevamente es producto del tratamiento de imágenes realizado en la muestra para eliminar interferencias en la imagen de tal manera que el objeto de estudio sea únicamente la hoja. Como producto de este tratamiento el color plomo se convierte en un color aún más claro lo que se vuelve prácticamente imperceptible para el algoritmo.

Como se menciona en el prefacio que hace referencia a las consecuencias de la deficiencia de potasio, esta produce necrosis en los laterales de la hoja lo que visualmente se traduce en la muerte de la hoja, y si esta es muy avanzada, se tornara de color plomo.

La deficiencia de potasio es fácilmente perceptible para el ojo humano, pero debido al tratamiento de imagen realizado, se tiene como respuesta del algoritmo un falso negativo, esto ya que el área ocupada por la deficiencia tiene el color plomo casi en su totalidad. Ver fig 4-7.

Figura 4-7

Muestra con deficiencia de potasio con falso negativo.

La muestra posee una deficiencia de potasio, pero su necrosis aun no esta tan avanzada por lo que aún es posible ver la variación de color para el algoritmo y en este caso, el algoritmo será capaz de funcionar correctamente. Ver fig. 4-8.

Figura 4-8

Muestra con deficiencia de potasio de verdadero positivo

CONCLUSIONES Y RECOMENDACIONES

A continuación, se describen las conclusiones a las que se llegaron a partir del logro de los diferentes objetivos planteados para el desarrollo de interfaz gráfica de usuario para obtener la detección de deficiencias NPK en las hojas de mango.

Conclusiones

- Se recopiló la información necesaria acerca de la hoja de mango de tal manera que es posible conocer cuáles son las consecuencias visuales provocadas por las deficiencias existentes de nitrógeno, fósforo y potasio durante su cultivo, y así, obtener una tabla comparativa con la información necesaria en la que se detallan las consecuencias visuales necesarias para detectar las deficiencias antes mencionadas.
- La determinación de los algoritmos necesarios para el procesamiento de imágenes, el software de Anaconda, IDE de programación que es “Spyder” y las respectivas librerías que intervienen en el procesamiento, diseño y programación de la interfaz gráfica de usuario como son OpenCV y PyQt permitió la realización de un software libre con un procesamiento de imágenes correcto y funcional para la detección de colores y contornos lo que forma parte esencial de la detección de deficiencias a partir de las consecuencias visuales antes encontradas.
- La interfaz gráfica de usuario desarrollada, mediante los algoritmos de procesamiento de imagen y diseño, en el presente trabajo de investigación, permite al usuario cargar una imagen con muestras de hojas de mango y así conocer si alguna de sus hojas posee alguna deficiencia nutricional ya sea esta nitrógeno, fósforo o potasio, de forma visual.
- Las pruebas de funcionamiento realizadas a la interfaz gráfica de usuario y a los diferentes algoritmos de procesamiento para la detección de deficiencias, permitieron detectar condiciones necesarias para la adquisición de muestras, así como realizar las

respectivas correcciones dentro del algoritmo. De esta manera, se logró validar el algoritmo de cada deficiencia, teniendo que el algoritmo de detección de deficiencia de nitrógeno tiene una eficacia del 74,2% y una especificidad del 96,5%.; el algoritmo de detección de deficiencia de fósforo tiene un 73,7% para eficacia y 78,3% de especificidad; mientras que el algoritmo de detección de potasio tiene 64,1% para eficacia y 85,9% de especificidad; comprobándose el correcto funcionamiento de la interfaz gráfica.

Recomendaciones

- La mejor opción para la creación de sistemas libres es el uso de lenguajes de programación abierto de tal manera que el producto final resultante se pueda ejecutar sin restricciones.
- Es necesario seguir normas o criterios de uso cuando se desarrolla un sistema o interfaz gráfica de usuario, para que esta sea intuitiva, de fácil compresión y manipulación. En este caso, se debe usar la mínima cantidad de texto de tal manera que no exista sofocación en el usuario.
- Es importante que se verifique el funcionamiento respectivo a cada elemento que se añade o que es parte de la interfaz de usuario, de tal manera que el funcionamiento sea el que se requiere, siempre teniendo en mente las posibles acciones que pueda realizar el usuario y se pueda asegurar su funcionamiento sin ningún inconveniente.
- Siempre tener en cuenta el ambiente controlado o máxima presencia de luz para la adquisición de imágenes, debido a que la falta de luz en las imágenes puede provocar falta de precisión y así, errores en el funcionamiento y consecuentemente también en los resultados.
- Para futuros trabajos de investigación se recomendaría enfocarse en la fundamentación de deficiencias nutricionales en otros cultivos que se produzcan en nuestro país.

REFERENCIAS

[1] D. Fuentes Diaz, E. Chapis Cabrera, y E. Chapis cabrera, “LAS BASES DEL CAMBIO DE LA MATRIZ PRODUCTIVA EN ECUADOR (2006-2016)”, *Univ. y Soc.*, vol. 9, núm. 2, pp. 313–318, 2019, [En línea]. Disponible en: http://scielo.sld.cu/pdf/rus/v11n3/2218-3620-rus-11-03-186.pdf.

[2] G. Guerrero, “La producción del mango ecuatoriano”, *Perspectivas*, pp. 8–15, 2018, [En línea]. Disponible en: https://perspectiva.ide.edu.ec/investiga/wp-content/uploads/2018/06/Perspectiva-Junio-2018_1-P.pdf.

[3] F. hondureña de investigación Agricola., “Recomendaciones prácticas para la fertilización del cultivo de mango”, en *Hoja Técnica No 3*, vol. 3, núm. 3, 2009, p. 4.

[4] J. Mora, J. Gamboa, y R. Elizondo, *Guía para el cultivo del mango*. San José, Costa Rica: Imprenta Nacional, 2002.

[5] C. N. de I. A. CENIAP, C. C. de investigaciones A. de la R. Nor-Oriental, y C. F. N. de I. A. M. de A. Y, “Evaluación del estado nutricional del Mango y el Aguacate y distribución radicular del mango cultivado en los suelos de las mesas orientales en Venezuela”, 2018. http://www.drcalderonlabs.com/Cultivos/Mango/Evaluacion del Estado Nutricional del Mango.htm.

[6] E. Del Castillo Huaccha, “Desarrollo De Un Sistema De Visión Artificial Para Realizar Una Clasificación Uniforme De Limones”, Universidad Privada del Norte, 2018.

[7] S. Carrato, S. Marsi, E. Medvet, F. A. Pellegrino, G. Ramponi, y M. Vittori, “Computer vision for the blind: A dataset for experiments on face detection and recognition”, *2016 39th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2016 - Proc.*, pp. 1206–1211, 2016, doi: 10.1109/MIPRO.2016.7522323.

[8] J. Piscoya, “SISTEMA DE VISIÓN ARTIFICIAL PARA APOYAR EN LA IDENTIFICACIÓN DE PLAGAS Y ENFERMEDADES DEL CULTIVO DE SANDÍA EN EL DISTRITO DE FERREÑAFE”, UNIVERSIDAD CATÓLICA SANTO TORIBIO DE MOGROVEJO, 2019.

[9] A. Gamonal, “Diseño de un sistema por visión artificial para determinar la calidad de mandarinas ””, Universidad Tecnológica de Perú, 2020.

[10] P. Modesto y H. Delgado, “El mango : Generalidades”, 2015.

[11] S. Arivazhagan y S. V. Ligi, “Mango Leaf Diseases Identification Using Convolutional Neural Network”, *Int. J. Pure Appl. Math.*, vol. 120, núm. 6, pp. 11067–11079, 2018.

[12] L. Cedeño, K. Coello, y M. Reinoso, “Estudio de factibilidad para la implementación de una microempresa comercializadora y producta de dulce de mango artesanal en el cantón de Macará”, Escuela Superior Politécnica del Litoral, 2009.

[13] B. Campos y E. Calderón, “El Análisis Foliar para el Diagnóstico Nutritivo de Plantaciones de Aguacate. Toma de Muestras”, *Málaga. Cons. Agric. Pesca y Desarro. Rural. Inst. Investig. y Form. Agrar. y Pesq.*, pp. 1–8, 2015.

[14] E. Prasetyo, R. D. Adityo, N. Suciati, y C. Fatichah, “Mango leaf image segmentation on HSV and YCbCr color spaces using Otsu thresholding”, *Proceeding - 2017 3rd Int. Conf. Sci. Technol. ICST 2017*, pp. 99–103, 2017, doi: 10.1109/ICSTC.2017.8011860.

[15] E. Sotomayor, “Desarrollo de mango (Mangifera indica L .) en almíbar a base de miel de abeja y Stevia .”, *Univ. CATÓLICA SANTIAGO GUAYAQUIL, Ecuador*, p. 97, 2018, [En línea]. Disponible en: http://repositorio.ucsg.edu.ec/bitstream/3317/10188/1/T-UCSG-PRE-TEC-CIA-29.pdf.

[16] A. Lima, “Python, descubrir los colores y deficiencias”, *Python*, 2021.

https://es.acervolima.com/category/python/.

[17] Plantix, “Deficiencia de Potasio”, 2019. https://plantix.net/es/library/plant-diseases/700002/potassium-deficiency.

[18] S. Sadeghian, “Síntomas visuales de deficiencias nutricionales en café”, *Cenicafe*, núm. Tabla 1, p. 12, 2017, [En línea]. Disponible en: www.cenicafe.org.

[19] Á. Romero Acero, A. Marín Cano, y J. Jiménez Builes, “Sistema de clasificación por visión artificial de mangos tipo Tommy”, *Rev. UIS Ing.*, vol. 14, núm. 1, pp. 21–31, 2015.

[20] H. Chen *et al.*, “Pre-Trained Image Processing Transformer”, pp. 12299–12310, 2020, doi: 10.1109/cvpr46437.2021.01212.

[21] C. E. Pardo, L. F. Sosa, E. A. Gutierrez, y F. R. Jiménez, “Classification system for blister pack of pills”, *2014 IEEE 5th Colomb. Work. Circuits Syst. CWCAS 2014 - Conf. Proc.*, 2014, doi: 10.1109/CWCAS.2014.6994614.

[22] M. E. González, “Técnicas de procesamiento de imágenes aplicadas al monitoreo de procesos alimentarios”, Universidad de la República de Uruguay, 2021.

[23] S. M. Audiovisuales, “Fotografía Digital”.

[24] C. E. Widodo, K. Adi, y R. Gernowo, “Medical image processing using python and open cv”, *J. Phys. Conf. Ser.*, vol. 1524, núm. 1, pp. 8–12, 2020, doi: 10.1088/1742-6596/1524/1/012003.

[25] G. Viera, “Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi para la clasificación del cacao”, Universidad de Piura, 2017.

[26] S. Gollapudi, “Learn Computer Vision Using OpenCV”, *Learn Comput. Vis. Using*

OpenCV, pp. 31–50, 2019, doi: 10.1007/978-1-4842-4261-2.

[27] R. Chino, “DISEÑO DE UN ALGORITMO DE PROCESAMIENTO DE IMÁGENES DEL SISTEMA DE PESAJE PARA EL CONTROL AUTOMÁTICO DE UNA FAJA TRANSPORTADORA EN LA UNIDAD MINERA MALLAY”, UNIVERSIDAD NACIONAL DEL ALTIPLANO, 2019.

[28] O. Rodríguez, “Desarrollo de una aplicación de reconocimiento en imágenes utilizando Deep Learning con OpenCV”, Universitat Politecnica de Valencia, 2018.

[29] ^ Ashok, K. Nanduri, R. Neelapu, V. Rao Maddumala, B. R. Domathoti, y K. Gowrisankar, “Image Recolorization With Convolutional Neural Network Using Opencv”, *J. Nat. Remedies*, vol. 21, núm. 4, pp. 81–87, 2020.

[30] G. Di Giussepe, “Detección de objetos por colores en imágenes con Python y OpenCV”, *Medium*, 2009. https://medium.com/@gastonace1/detección-de-objetos-por-colores-en-imágenes-con-python-y-opencv-c8d9b6768ff.

[31] “Espacio RGB”, [En línea]. Disponible en: https://www.hisour.com/es/rgb-color-space-24575/.

[32] “Modelos y códigos de espacio de color RGB, YUV y HSV”. https://programmerclick.com/article/24061863706/.

[33] A. V. Muñoz, *Principios de color y holopintura*, 1a ed. Barcelona, España, 2013.

[34] J. M. Willman, *Beginning PyQt*. 2020.

[35] Instituto Ecuatoriano de Normalización, “NTE-INEN ISO/IEC:25010 - Sistemas e Ingeniería de Software - Requerimientos y Evaluación de sistemas y calidad de software (SQUARE) - Modelos de calidad del sistema y software (ISO/IEC 25010:2011, IDT)”, 2011.

ANEXOS

Al tener líneas de código muy extensas se adjunta los archivos para la instalación de la interfaz gráfica de usuario desarrollada.

De igual manera se adjunta el manual de usuario dentro de la GUI desarrollada.