Portada

INDICE / Contenido

Agradecimiento/Dedicatoria (opcional) 1 c/u **esteban/robin**

Resumen **Esteban**

Palabras Claves **Esteban**

Capitulo 1 – Introduccion

Introducción **Robin**

Fundamentos **Robin**

Industria del limón, Exportación, Como debe ser el limón **Robin**

Objetivos **Esteban**

Capitulo 2 - Desarrollo

Software utilizado (Descripción General) **Esteban**

Python

Opencv

PyQt

Qt Designer

Visual Studio Code

Diagrama general del proyecto compuesto por cámara-pc(Python)-esp-32- servomotor-display-led

Hablar de cada componente **Robin**

Camara Web

Iluminacion

Esp-32

Servomotor

Display

Software Implementado

Interfaz gráfica **Robin**

Captura de Imagen / Video **Esteban**

Selección de área para analizar **Esteban**

Detección de color **Esteban**

Encontrar centro y diámetro **Esteban**

Contador de limones **Esteban**

Diagrama de flujo

Hardware Implementado

Control por serial **Esteban**

Manejo de servos – código **Esteban**

Código para display **Esteban**

Plaqueta del Proyecto

Cuarto de visión **Robin**

Capítulo 3 Resultados

Capítulo 4 Análisis de costos

Capítulo 5 Conclusiones

Anexos

Instalación de Python y librerías

Códigos Usados

Bibliografía

# Resumen

En el presente proyecto se realizó un sistema que utiliza el procesamiento de imágenes para la clasificación automática de limones.

Para llevarlo a cabo se utilizó el lenguaje de programación Python, junto a la librería OpenCV, con los cuales se procesan las imágenes procedentes del cuarto de visión, conformado por una web cam y el sistema de iluminación, el resultado del procesamiento es enviado al microcontrolador esp-32 mediante comunicación serial para que se encargue del accionamiento de los servomotores.

Se utilizó el entorno de desarrollo Visual Studio Code para el desarrollo del programa de PC, y QtDesigner para el desarrollo de la interfaz gráfica.

Una vez finalizado se logró clasificar los limones entre los que son aptos para exportación, los que son para consumo local y los que son descartados, conocer el diámetro de cada limón y calcular cuántos y que cantidad de kilogramos son procesados.

**Palabras Clave:** Clasificación de limones, procesamiento de imágenes, Python, OpenCV

# Introducción

\*Para introducción hablar un poco sobre el limón, clasificación, innovación, etc.

# Objetivos

## Objetivo General

Diseñar y construir un sistema clasificador de limones, implementando la selección de color y tamaño por medio de visión artificial para la distinción del fruto tipo exportación.

## Objetivos Específicos

* Diseñar y construir un sistema que permita la clasificación de limones para seleccionar el limón adecuado según la norma de calidad internacional.
* Diseñar un programa de adquisición de imágenes que permita la clasificación del limón por el color y tamaño.
* Diseñar la interfaz gráfica (GUI) que permita ver la información resultante.
* Diseñar los circuitos de control para cada uno de los actuadores del sistema.
* Construir un prototipo funcional que permita realizar pruebas de funcionamiento de los sistemas que clasifican el fruto.

# Capítulo 2 – Desarrollo

## 2.1 Software Utilizado

### 2.1.1 Python

Es un lenguaje de programación interpretado de alto nivel, interactivo y orientado a objetos. El código fuente está disponible bajo Licencia Publica General de GNU. La sintaxis de Python es muy limpia, con énfasis en la legibilidad y utiliza palabras clave estándar en inglés.

Python fue creado a finales de la década de los años 80 por un europeo llamado Guido Van Rossum en los Países Bajos. El objetivo de Guido era cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en Unix usando C.

Su formato no utiliza corchetes para delimitar bloques, sino que son las tabulaciones y los espacios los que definen en que bloque se encuentra una instrucción. Una de sus características es el uso de palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos ‘!’, ‘||’ y ‘&&’ en Python se escriben ‘not’, ‘or’ y ‘and’, respectivamente.



*Figura 1 – Logo de Python[[1]](#footnote-1)*

## 2.1.2 Librerías

### 2.1.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de maching learning y visión por computadora de código abierto. Al ser un producto con licencia BSD, OpenCV facilita a las empresas a utilizar y modificar el código.

La biblioteca fue lanzada oficialmente en 1999 y posee más de 2000 algoritmos optimizados que se pueden utilizar para detectar y reconocer colores, caras, identificar objetos, clasificar acciones humanas en videos, rastrear movimientos de cámara, etc.

Es actualmente utilizado por grandes compañías como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda y Toyota.



*Figura 2 – Logo de OpenCV[[2]](#footnote-2)*

### 2.1.2.1 PyQT

Es una biblioteca que nos permite la conexión entre la interfaz gráfica desarrollada en el software QT Designer y el código desarrollado en Python.

## 2.1.3 QT Desingner

Es una herramienta para diseñar y crear interfaces gráficas de usuario (GUI), permite componer y personalizar ventanas y cuadros de diálogos de manera *“what-you-see-is-what-you-get”* (Lo que vemos es lo que optenemos).



*Figura 3 – Logo de QT[[3]](#footnote-3)*

2.1.4 Visual Studio Code

Es un potente editor de código gratuito y de código abierto, desarrollado por Microsoft y lanzado en 2016, mediante sus extensiones permite convertirlo en un entorno de desarrollo que soporta múltiples lenguajes de programación como Python, Java Script, C, etc. Entre sus características más destacadas se puede mencionar la finalización de código inteligente, que nos permite agilizar la escritura de código, depuración optimizada, resaltado de sintaxis, entre otras.



*Figura 4 – Logo de Visual Studio Code[[4]](#footnote-4)*

## 2.4 Software Implementado

### 2.4.1 Captura de Imágenes

En este punto nos centraremos en la captura de imágenes provenientes de la cámara para su posterior análisis y clasificación.

Para nuestro propósito el código hace uso de la librería OpenCV, la cual mediante la función *VideoCapture(0)* inicia la cámara para tomar una imagen (0 es el número de cámara en caso de tener más de una conectada), dicha imagen se almacena en una variable y la función *read* nos devuelve el frame capturado.





*Figura 5 – Captura de Imagen*

2.4.2 Captura de Video

Primer recordemos que un video no es más que un conjunto de imágenes en un determinado tiempo, generalmente 30 por segundo. Teniendo en cuenta esto para realizar la captura continua de video se pondrá las funciones anteriormente mencionadas en un siclo while infinito que permite la lectura continua de imágenes.



2.4.3 Selección del área a analizar

Utilizando la función *np.array,* la cual pertenece a la librería Numpy y nos permite crear arreglos de números, definiremos 4 puntos que ordenados de forma horaria formaran nuestra región de interés sobre la cual aplicaremos posteriormente las funciones que nos permitirán realizar el procesamiento.

****

2.4.4 Detección de Color

Para realizar la detección de color lo primero que se debe realizar es la conversión del espacio de color.

Por defecto OpenCV carga las imágenes en un espacio de color BGR (Blue, Green, Red), es necesario transformar las imágenes a un espacio HSV (Hue, Saturation, Value / Matiz, Saturación, Brillo).

Utilizaremos el espacio HSV debido a que nos permite identificar de forma más sencillas los colores. Para determinar un color nos centraremos en el componente H que corresponde a la Matiz, en la siguiente imagen podemos apreciar que el componente H va desde el rojo, pasando por el verde y azul hasta llegar al rojo nuevamente.



Figura 5 – Espacio de color HSV [[5]](#footnote-5)

Además, para trabajar con estén entorno de color es necesario saber que valores puede tomar cada componente en OpenCV:

H: 0 a 179

S: 0 a 255

V: 0 a 255

La función que nos permite realizar la transformación del espacio de color es *cv2.cvtColor* la cual recibe 2 parámetros primero la imagen a transformar el segundo entre que espacio de color se realizará la trasformación.

****

* Determinar los rangos donde se encuentra los colores a detectar

Los colores que nos interesan detectar son el amarillo y verde, los cuales son los colores característicos del limón, la presencia de otros colores como podría ser el marrón correspondería a alguna anomalía.



Figura 6 – Componentes HSV [[6]](#footnote-6)

Como se observa en la figura 6 el componente H va desde 0 a 179, luego de diferentes pruebas se llegó a determinar los siguientes rangos:

Para el color Amarillo el rango en H va de 22 a 33, el de S va de 21-255 y V 0 a 255.

Para el color Verde el rango en H va de 35 a 89, el de S va de 50-255 y V 0 a 255.

Para Defecto el rango en H va de 0 a 20, el de S va de 150-255 y V 80 a 255.

Estos valores correspondientes a los diferentes rangos de colores son creados como arreglos de números mediante la función *np.array* y almacenados en variables.

****

Para buscar los rangos de colores en nuestra imagen se utiliza la función *cv2.inRange,* la cual recibe como parámetros la imagen en la cual se buscará los rangos, y los arreglos correspondientes al límite inicial y final del color que nos interesa.

****

* Consideraciones que se deben tener en cuenta

Se debe tener en cuenta dos aspectos muy importantes a la hora de aplicar la detección de colores:

**Iluminación:** es importante tener controlado este aspecto ya a que a más o menos iluminación presente sobre el objeto, puede variar el resultado de la detección.

**Fondo:** el fondo también es un aspecto a tener en cuenta debido a que si en este se encuentran presentes alguno de los colores dentro de los rangos no solo se detectaría nuestro objeto de interés sino también será detectado parte del fondo. Se recomienda en lo posible usar un fondo uniforme y que no presente ningún color de interés.

2.4.5 Detección de contornos

Una vez que se tiene identificado los grupos de pixeles que nos interesan mediante la función *cv2.findCountours* encontramos los contornos que forman el área de estos grupos, luego es necesario analizar uno a uno los contornos para asegurarnos que se traten de limones. Los contornos de un objeto se pueden explicar simplemente como una curva que une todos los puntos continuos (a lo largo del límite o borde) que tienen el mismo color o intensidad.

****

Procederemos a analizar cada contorno contenido en la variable contornos, compararemos si el área del contorno es lo suficientemente grande para ser considerado un limón. Para ellos se compara el área en pixeles con 1500, de modo que si el área es mayor a este valor se continua con el procesamiento de lo contrario lo descartaremos.



2.4.6 Encontrar Centro y diámetro

Al aplicar la función *cv2.boundingReact* podemos encontrar 4 puntos útiles, las coordenadas X, Y y los valores W y H que corresponden al ancho y altura respectivamente. Estos valores nos serán útiles después para poder encontrar el diámetro, contar y dibujar un rectángulo al contorno de nuestro objeto.



Manteniendo fija la distancia de la cámara al objeto, para hallar el diámetro de cada limón se calcula el promedio del ancho (W) y la altura (H). Este dato esta expresado en función de numero de pixeles, para convertirlo en milímetros lo dividiremos por un factor. Este factor lo estableceremos mediante pruebas, se calculará el diámetro mediante el software y se comparará el valor con una medición real del limón en milímetros, con el promedio de todas las mediciones realizadas se establecerá una relación que nos permite definir el factor.



**Tabla conm los valores dem muestras y medidas con calibre**

Dándole los parámetros obtenidos con boundingRec a la función *cv.rectangle* obtendremos un rectángulo al contorno de nuestro objeto, este rectángulo nos será útil para entender mejor el proceso de conteo de limones.

2.4.7 Contador de Limones

Para poder contar los limones, tomaremos el punto “Y” del recuadro delimitador antes mencionado y veremos si está presente entre 300 y 400, si es así el limón será contado haciendo que la línea amarilla central se torne verde y las variables destinadas a llevar las cuentas de los limones aumentaran.



Es importante mencionar que los contadores podrían fallar si un limón se desplaza demasiado lento, lo ideal sería que la velocidad se mantenga constante. Luego de pruebas y error se llegó a determinar cuál sería la velocidad óptima para los rangos determinados.

Tanto el rectángulo del contorno como la línea verde del contador son solamente con fines visuales para una mejor apreciación del proceso, podrían ser excluidas del código sin modificar su funcionamiento.

1. https://www.python.org/ [↑](#footnote-ref-1)
2. https://opencv.org/ [↑](#footnote-ref-2)
3. https://www.qt.io/ [↑](#footnote-ref-3)
4. https://code.visualstudio.com/ [↑](#footnote-ref-4)
5. <http://omes-va.com/deteccion-de-colores/> 17-09-2022 [↑](#footnote-ref-5)
6. <http://omes-va.com/deteccion-de-colores/> 17-09-2022 [↑](#footnote-ref-6)