DETERMINACIÓN DEL ESTADO DE MADUREZ DEL AGUACATE MEDIANTE PROCESAMIENTO DE IMÁGENES CON LA RASPBERRY PI.

Miguel Ángel Escobar Peláez

José Orlando Castaño García

Proyecto de grado presentado como requisito parcial para optar por el título de Ingeniero Electricista

Director:

Luis Hernando Rios Gonzales, PhD (c).

Universidad Tecnológica de Pereira

Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación

Programa de Ingeniería Eléctrica

Pereira

2018

Agradecimientos

Quiero agradecerle a mi madre
A mi hermana y amigos
Por acompañarme en esta travesía,
Su apoyo fue esencial para que esto fuera posible.
- *Miguel A. Escobar*

Quiero agradecer a Dios, a mi Padre, Madre y Hermanos
Por el apoyo que recibí de ellos en todo momento,
A Silvana y amigos que siempre me apoyaron en este proceso.
- *Jose O. Castaño*

Queremos agradecer a nuestro director de tesis
Por su guía y orientación,
Su enseñanza iluminó el camino de este proyecto de grado.
Muchas Gracias.
- *Miguel A. Escobar y Jose O. Castaño*

Contenido

1. Resumen

Este proyecto se basa en el procesamiento de imágenes para determinar el estado de madurez de frutas, en este caso se utilizará el aguacate, la determinación se hará mediante la captura de una imagen, de la cual se analizarán diferentes características que posee la fruta, de la cuales se tomará como principal su *color promedio*. Posteriormente, la clasificación se hará mediante el uso del algoritmo de máquinas de vectores de soporte (SVM) con la ayuda de la librería de OpenCV en Python y la Raspberry Pi como sistema embebido.

Los sistemas de control de calidad en la industria agro tienen grandes avances gracias a las técnicas de procesamiento digital de imágenes, la idea que se permite entrar en contexto se basa en el análisis realizado sobre las capturas tomadas bajo un ambiente controlado, este se emplea para la captura de la imagen del aguacate, y analizando aspectos de la fruta como lo son la forma, tamaño y color, mostrando su estado de maduración por medio del método de visión artificial.

2. Introducción

La determinación del estado de maduración de las frutas por medio de dispositivos computacionales o también llamada visión artificial se está volviendo fundamental debido a las ventajas que presenta para identificar diferentes características a objetos de diferentes formas, colores y tamaños.

Estas ventajas ofrecidas por estos dispositivos se emplean debido a su rapidez y la exactitud con la cual arrojará los resultados mediante la detección y la clasificación de aspectos de interés de la fruta obtenidos mediante la visión artificial.

Debido a que hoy en día los productores de aguacates realizan la clasificación de la fruta por medio de experiencia de expertos, basándose en su capacidad de observación, y esto en muchas ocasiones conllevan a problemas y presenta resultados poco confiables, y es debido a esto que se crea un algoritmo el cual permita hacer mediante un procesamiento de imágenes determinar ciertas características del aguacate, y con un análisis computacional empleando la Raspberry Pi B2 y la Pi Cam, poder determinar el estado de maduración en tiempo real.

3. Preliminares

3.1. Planteamiento del problema

Debido al avance tecnológico de los últimos años se ha permitido llevar todas estas tecnologías de visión artificial a diferentes áreas de la industria, una de estas es la de la exportación de frutas y en la actualidad la mayoría de clasificación de estas frutas se hacen manualmente y utilizando un buen observador y conocedor de estas. No es fácil a simple vista ver las cualidades que presenta el aguacate y es preciso tener un contacto de forma tangible para determinar el estado de madurez en el que se encuentra, y esto conlleva a que el aguacate se vea sometido a alteración de su integridad.

Debido a todas estas consideraciones se desarrolla este proyecto en pro de un manera más eficiente y confiable de determinar el estado de madurez de esta fruta, y poder llevar las tecnologías a estos campos y tener un mayor control en cuanto la calidad del producto lo permita.

3.2. Justificación

En los últimos años, el deseo de automatizar nuestro entorno ha tomado un papel importante debido a las necesidades que presentan en él. Y ya que algunos sistemas electrónicos embebidos son dispositivos económicos y versátiles que permiten ser utilizados para la programación de operaciones especificas y se utilizan estos al servicio de la industria para poder llevar tareas específicas de forma rápida y confiable para hacer uso de los datos adquiridos para tomar decisiones ya sean del interés deseado.

Es de necesidad llevar estas tecnologías de visión artificial a todos los campos de la industria para aprovechar todo su potencial, y debido a que en el campo de la exportación de frutas se desea tener la más alta calidad en lo posible, se pretende aislar todo contacto, y es por esto por lo que se emplea la visión artificial como opción principal para determinar las características necesarias en el estado de maduración del aguacate y crear un ambiente amigable para la correcta lectura de los datos adquiridos.

3.3. Objetivos

3.3.1. Objetivo general

Diseñar un algoritmo que permita determinar el estado de maduración del aguacate mediante el procesamiento digital de imágenes, empleando la Raspberry pi B 2 y la Pi Cam, en ambiente controlado.

3.3.2. Objetivos específicos

Estudiar los diferentes métodos de clasificación que existen para determinar características extraídas del aguacate.

Implementar una interfaz grafica que permita mostrar el aguacate en el estado de madurez en el que se encuentra.

4. Marco teórico

Los estados de maduración que se presentan normalmente en el aguacate se clasifican de las siguientes formas:

Madurez fisiológica: la fruta se encuentra fisiológicamente madura cuando ha logrado un desarrollo normal y posee un tamaño en el cual puede continuar madurando después de estar cosechado.

Madurez hortícola: en este estado la fruta se encuentra en un estado apto para el consumo final.

Madurez organoléptica o de consumo: es este estado la fruta tiene características (color, sabor, aroma, textura) para su consumo.

Climaterios: son aquellos frutos los cuales continúan el proceso de maduración después de ser cosechados.

Procesamiento de imágenes: Es el conjunto de las técnicas que se aplican a las imágenes digitales con la finalidad de mejorar la calidad o la extracción de información de interés.

4.1. Hardware

La Raspberry pi B2 es un sistema embebido que tiene como características una placa de 8.5 por 5.3 cm, un chip integrado Broadcom BCM 2935, que contiene un procesador ARM11 con varias frecuencias de funcionamiento y la posibilidad de subirla hasta 1 GHz, un procesador gráfico VideoCore IV y 512MB de memoria RAM, cuenta con una salida de video y audio a través de un conector HDMI, con lo que conseguiremos conectar la tarjeta tanto a televisores como a monitores que tengan la misma conexión, también cuenta con una salida de video compuesto y audio a través de un puerto minijack. Posee una conexión ethernet 10/100, una conexión Wifi, y 4 puertos USB.

La Raspberry pi B2 no es compatible con los sistemas operativos Microsoft Windows y MAC, pero permite instalar versiones del sistema operativo Linux. Este sistema embebido no posee disco duro por lo que es necesario utilizar una tarjeta de memoria SD, posee puertos USB para conectar periféricos como el ratón, teclado o un receptor de red Wifi. Diseñado en el Reino Unido por la fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. Raspberry Pi cuenta actualmente con cinco modelos diferentes: Raspberry Zero, Raspberry 1 modelos A+ y B+, basados en Raspberry Modelos Ay B, Raspberry Pi 2 modelo B, Raspberry Pi 3 Modelo B.

El módulo de la cámara empleada en este proyecto para la captura de imágenes cuenta con 5 megapíxeles y está diseñado específicamente para la Raspberry Pi con un lente de foco fijo, es capaz de tomar imágenes estáticas de 2592 x 1944 y también es compatible con el formato de video 1080p30, 720p60 y 640p60/90. Se conecta a la Raspberry Pi por medio de un pequeño conector en la parte superior de la tarjeta y utiliza la interfaz dedicada CSi, diseñado especialmente para la conexión de cámaras. La placa en sí es muy pequeña, alrededor de 25 mm por 20 mm por 9mm, y pesa un poco más de 3 gramos, por lo que lo hace perfecto para aplicaciones donde el tamaño y el peso son importantes.

Raspberry pi 2 modelo B

Se describe principalmente la Raspberry Pi 2 modelo B, en base que es el sistema embebido utilizado junto con la Picamara, las cuales son de la misma familia.

Figura 1. Raspberry Pi 2 model B

Figura 2. Cámara para Raspberry Pi 5MP

Puertos USB

Este modelo posee dos puertos USB, permitiendo conectar teclado y ratón, dispositivos de almacenamiento externo USB, Wifi USB, entre otros.

Puerto ethernet

Incluye puerto Ethernet para conectar la Raspberry a red cableada, lo que permite tener un acceso a Internet y hace posible que otros dispositivos en la red puedan acceder a la Raspberry Pi.

Conector CSI (camera serial interface)

Es un conector tipo bus de 15 pines utilizado para añadir un dispositivo compatible con la interfaz CSI-2 (Camera Serial Interface versión 2). Donde es posible conectar la cámara de la Raspberry Pi.

Conexión a pantalla

Raspberry Pi es compatible con tres salidas de video diferentes: video compuesto, video HDMI y video DSI. El video DSI requiere de un hardware avanzado para poder ser utilizado, mientras que el video compuesto y video HDMI son más accesibles para el usuario.

Conexión de audio

No es una conexión necesaria si se conecta una pantalla por medio del puerto HDMI ya que este es capaz de transportar las dos señales tanto la de video como la de audio, si se está usando la salida de video compuesto la Raspberry solo estaría transportando video por lo que si se desea la señal de audio analógica estará disponible una ranura de audio Jack de 3.5 mm.

Almacenamiento

Raspberry Pi viene por diseño sin disco duro por lo que cuenta con una ranura para memorias SD, casi cualquier tarjeta SD funcionará con Raspberry Pi. Está tarjeta SD debe ser de al menos 2 GB ya que será la que albergará todos los archivos requeridos por el sistema operativo.

Alimentación

Raspberry Pi se enciende automáticamente con la conexión a la fuente de alimentación por lo que no posee botones de encendido y apagado. Para su alimentación se dispone de un conector micro USB que suministra 5 Voltios y mínimo 0.7 Amperios.

Pines GPIO (general inputs and outputs)

El diseño de los pines del puerto GPIO de Raspberry Pi varía ligeramente según el modelo de placa que se posea. Todos los modelos actuales más recientes cuentan con el mismo diseño de pines. Antes de utilizar el puerto GPIO, se debe conocer el modelo de Raspberry Pi que se está utilizando.

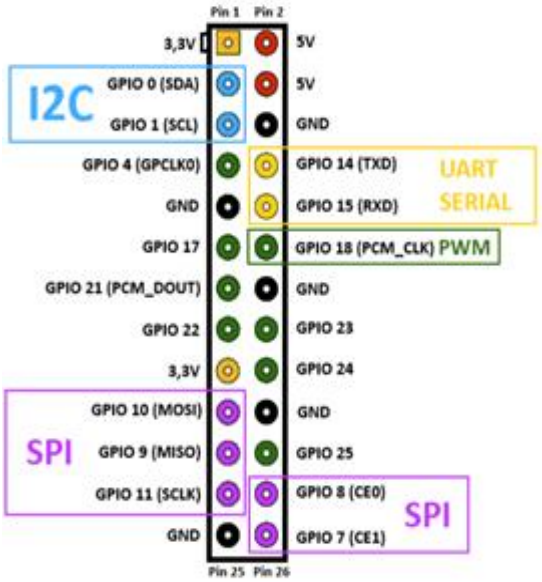


Figura 3. Diagrama puerto GPIO Raspberry Pi 2 model B

Decodificador

El decodificador es un dispositivo que acepta una entrada digital codificada en binario y activa una salida. Este dispositivo tiene varias salidas, y se activará aquella que establezca el código aplicado a la entrada.

Sistema Operativo Rapsberry PI

El sistema operativo de Raspberry Pi debe ser cargado en la memoria SD, esto se hace descargando una imagen del sistema operativo desde la página de Raspberry Pi. Para este proyecto se usó la distribución RASPBIAN JESSIE.

Linux

El sistema operativo de Raspberry Pi es el GNU/Linux, como Linux es de código abierto lo que quiere decir que se puede descargar el código fuente del sistema operativo y hacer los cambios que se deseen. Esta forma de desarrollo es lo que ha permitido a Linux ser modificado para ejecutarse en la Raspberry Pi.

Debian

El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo (SO) libre. Los sistemas Debian actualmente usan el núcleo de Linux o de FreeBSD. Linux es una pieza de software creada en un principio por Linux Torvalds y desarrollada por miles de programadores a lo largo del mundo. FreeBSD es un sistema operativo que incluye un núcleo y otro software.

Python

Es un lenguaje de programación cuya sintaxis es muy simple, clara y sencilla. Otra de las razones para usar Python es la compatibilidad con Raspberry Pi, ya que puede interactuar con los pines del puerto GPIO del dispositivo simplemente llamando las funciones codificadas.

4.2 La librería de Open CV

OpenCV es una biblioteca de software con soporte para C++, Java, Python y MATLAB con interfaces en Windows, Linux, Android y Mac OS. Es una librería de código abierto especializada en visión por computadora y machine learning que ha sido utilizada por compañías como Google, Microsoft, Sony y Honda en proyectos de navegación robótica, detección de acciones humanas en video, monitorización de equipo en procesos automáticos y mucho más.

La librería tiene más de 2500 algoritmos optimizados que se pueden utilizar para hacer reconocimiento de rostros, identificar objetos, clasificar acciones humanas en video, y hacer seguimiento de objetos. Este proyecto utilizará en concreto algunas de sus funciones para hacer procesamiento de imágenes en la detección de bordes de un objeto (fruta situada en el centro de la imagen) y utilizar el algoritmo de SVM (support vector machine) para hacer la clasificación por machine learning.

4.3 Detección de Bordes

La detección de bordes a menudo en OpenCV se utiliza con el algoritmo Detector de bordes Canny, el cual consta de tres partes: primero se usa el algoritmo de detección de bordes Sobel (se basa en el cálculo de la primera derivada para detectar cambios de intensidad), después se hace una supresión de píxeles fuera del borde y por último se aplica umbral por histéresis. Sin embargo, para el caso del aguacate se puede obviar los dos primeros pasos del algoritmo como se demostrará más adelante en el desarrollo del trabajo.

El umbral por histéresis, tercer paso en el algoritmo de detección de bordes Canny, es el método más simple de segmentación de una imagen. Consiste en binarizar la imagen por intensidad de su pixel (la imagen debe estar en escala de grises) y por su relación respecto un valor de umbral.

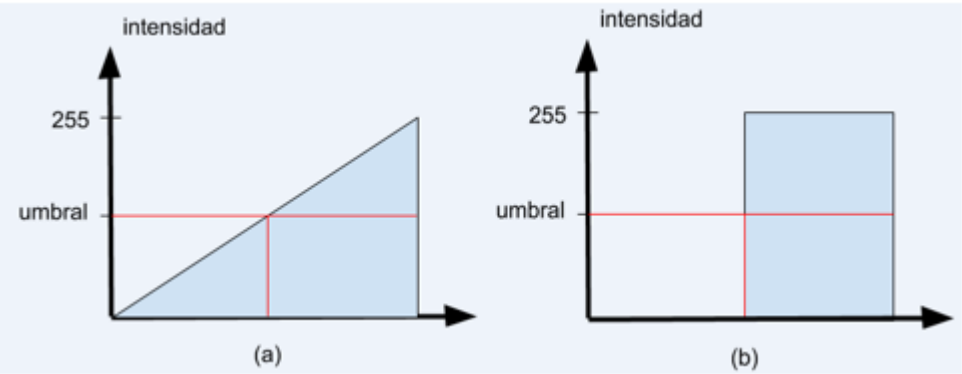


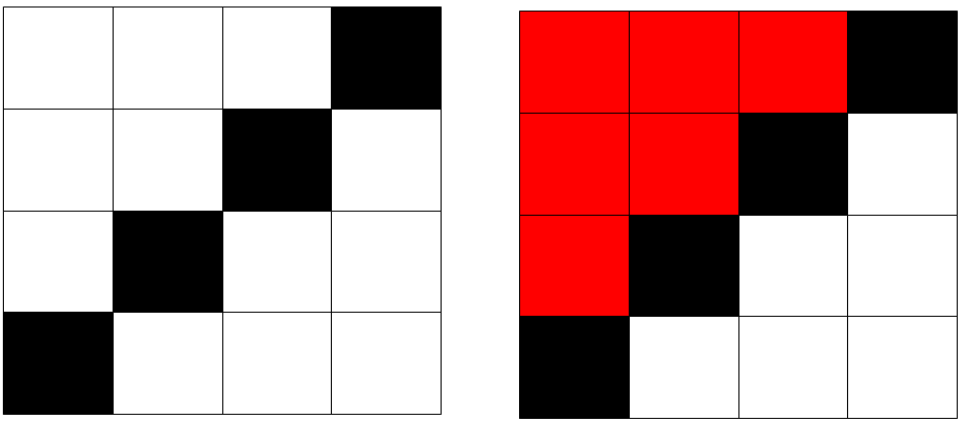
Figura 4. a) Imagen antes de aplicar umbral por histéresis, b) Umbral por histéresis aplicado.

Como se observa en la Figura 4 (a), la imagen corresponde a las intensidades de gris organizadas desde 0 hasta 255 (de negro a blanco), y después de implementar el umbral por histéresis (Figura 4 (b)) los pixeles arriba de la referencia se tornan blancos y por debajo negros.

4.4 Floodfill

El algoritmo de relleno por difusión floodfill, creado por S. Fazzini, es utilizado en muchos programas de edición de imágenes como Adobe Photoshop y GIMP para rellenar áreas y secciones de una foto con un color determinado. Fue creado

La función de floodfill en OpenCV requiere de tres parámetros de entrada después de la imagen: una máscara (opcional), un punto inicial, y un color para hacer el relleno. Funciona buscando todos los pixeles contiguos al punto inicial y a ellos mismos que no hagan parte de la máscara utilizada o de un color diferente al del punto inicial. En la figura 5 a y b se puede ver la implementación del algoritmo floodfill en una imagen sencilla de 4x4 pixeles inicializando en el primer pixel y rellenando con rojo.



(a) (b)

Figura 5. a) Imagen de 4x4 pixeles.
b) Floodfill en el punto (0,0).

4.5 SVM

La máquinas de vectores de soporte o SVM (por las siglas en inglés de support vector machine) constituyen un método lineal para la clasificación de elementos. Hacen parte del campo de aprendizaje de máquinas o aprendizaje artificial, es decir, es un algoritmo que es capaz de generalizar comportamientos basados en unos ejemplos de entrenamiento. A diferencia de otros algoritmos de aprendizaje que tratan de minimizar un error, las máquinas de vectores de soporte se basan en minimizar un riesgo estructural.

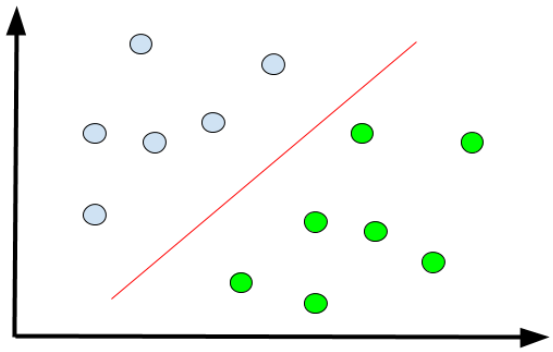


Figura 6. Máquina de vectores de soporte.

Como se puede ver la figura 6, el objetivo del algoritmo es crear espacios de clasificación separados por un hiperplano que maximiza la distancia entre los ejemplos más cercanos de cada clase o categoría de los elementos.

5. Estado del arte

El procesamiento de imágenes durante los últimos años se convirtió en una técnica especial para la determinación y clasificación más utilizadas en lo que la automatización de procesos de todo tipo de la industria se refiere ya que sus ventajas sobre la identificación de patrones sobre imágenes permitiendo extraer información importante ya sea el tipo de aplicación.

Dado que el caso de identificación de la madurez de las frutas depende de atributos como lo son el color, forma, sabor y textura, es fundamental realizar un control a partir de atributos internos.

A continuación se mencionan algunos trabajos previos sobre lo que tiene que ver con la determinación del estado de madurez y otros aspectos de interés.

EVALUACIÓN DEL PROCESO DE REHIDRATACIÓN DEL LIOFILIZADA DE AGUACATE CRIOLLO(persea, americana mill. variedad Drymifolia) MEDIANTE ANALISIS DE IMÁGENES.

Elaborada por Wendy Dayanara Acosta Hernandez, en el cual evalúa el proceso de humectación del liofilizado de aguacate criollo para determinar el grado de humectación y hidratación del polvo liofilizado a través de análisis de imágenes por mojado directo. Como materia prima utilizo pasta de aguacate variedad criollo. los equipo utilizados para este proyecto son: una cámara digital modelo L110 de 8.2 megapixeles marca samsung, microscopio óptico nikon SMZ 1500, japan, termobalanza, brainweigtj, modelo MB300, usa. liofilizadora labconco, modelo 7751000, entre otros.

DISEÑO E IMPLEMENTACIÓN DE UNA HERRAMIENTA COMPUTACIONAL PARA LA IDENTIFICACIÓN DEL ESTADO DE MADUREZ DE LA GRANADILLA MEDIANTE TÉCNICAS DE VISIÓN ARTIFICIAL SOBRE UN ORDENADOR DE PLACA REDUCIDA.

Presentada por Diego Albéniz Figueroa, donde parte desde la cosecha para una fruta de tipo exportación y se analizara en el caso particular de la granadilla son consideradas como maduras aquellas que tienen su cáscara de color amarillo, son consideradas como verdes aquellas donde su cáscara aún es verde y son consideradas como muy maduras o dañadas aquellas donde su color se torna oscura. Esta prueba consiste en una inspección visual por parte del técnico experto, donde se realizan pruebas que dependen de características físicas como: el color de la corteza, dimensiones, firmeza y presencia de hojas secas en el árbol, características químicas como sólidos solubles, pH y acidez y finalmente, características organolépticas como sabor, aroma, textura y color según los criterios establecidos por el Instituto Colombiano Agropecuario (ICA) y manual de manejo cosecha y pos-cosecha de granadilla (Corporación Colombiana de Investigación Agropecuaria CORPOICA, 2008).Esta investigación pretende dar un soporte tecnológico, flexible y portabilidad a un sistema que permita reducir la subjetividad en las mediciones del técnico experto en el estado de madurez de granadillas, a partir de técnicas de procesamiento de imágenes.

PROCESAMIENTO DE IMÁGENES PARA LA CLASIFICACIÓN DE CAFÉ VERDE.

Presentada por Manuel Alejandro Arias y Jorge Antonio Sierra Ruiz, cuyo objetivo en este proyecto es el del desarrollo de un algoritmo por el cual permite la detección de defectos en el café verde utilizando el procesamiento de imágenes. El procesamiento en este proyecto se basó mediante la herramienta de MATLAB ya que ofrece mayor facilidad que OpenCV en diferentes aspectos como lo son la manipulación de imágenes, que las variables no dependen del tipo de declaración y por último que consideran que tiene herramientas más interactivas para evaluar el desempeño del sistema mediante la comparación de una pre-selección manual de café verde y lo que el sistema clasifica.

PROCESAMIENTO DE IMÁGENES USANDO OPENCV APLICADO EN RAPSBERRY PI PARA LA CLASIFICACION DEL CACAO.

presentado por Gabriela Viera Maza, en la cual presenta un proyecto para desarrollar en el cual busca establecer los conceptos básicos para el desarrollo de la visión artificial para la clasificación de granos de cacao según sus características externas como el tamaño en la fase final del secado. Por otra parte, los ambientes controlados para estas prácticas radican generalmente en la Cámara y unos sensores.

DETECCION Y CLASIFICACION DE DEFECTOS EN FRUTAS MEDIANTE EL PROCESAMIENTO DIGITAL DE IMÁGENES.

presentado por Leonario Pencue Fierro y Jauri León Téllez, donde por medio del análisis digital de imágenes en frutas, en total 165 frutos para la realización del entrenamiento y validación del sistema, se tuvo en cuenta la validación del sistema según la norma NTC4086 del ICONTEC que está de acuerdo con la calidad visual que busca el consumidor, los tipos de daños que buscaban principalmente por medio del análisis digital de imágenes eran principalmente las manchas, cicatrices, lesiones oscuras, podredumbre, y también analizar el buen estado de la madurez la forma y el tamaño, clasificándolo en 4 clases, siendo la ultima el estado de desecho de la fruta.

La adquisición de las imágenes la realizaron mediante una cámara CCD JVC Modelo TK-C1380 a color, y en específico de las 165 especies eligieron la Naranja valencia para aplicar los análisis, estas se colocan en una cámara de iluminación blanca, y las imágenes se digitalizaron mediante una tarjeta Matrox Meteor II/Std y se procesaron mediante una estación Leica Q5501W, el software para la adquisición de datos y el procesamiento de las imágenes fue por medio de Qwin/Quips.

RECONOCIMIENTO DE OBJETOS UTILIZANDO OPENCV Y PYTHON EN UNA RAPSBERRY PI 2 EN UNA TLAPALERIA.

Presentada por Guadalupe Jonathan Gonzales Osorio, donde que un problema que se presenta en las tlapalerías y Negocios similares es que un cliente entra con una pieza de mostrario que desea comprar, y el vendedor no conoce ni el nombre y las especificaciones de la pieza, entonces esto obliga a una búsqueda por medio de todas las piezas que tiene el almacén, haciendo un análisis comparativo con lo que se

encuentra en su base de datos, evitando así una pérdida de tiempo si tocara hacer a un operario.

Entonces con la implementación de un algoritmo que permita el reconocimiento digital de imágenes y así permitir brindar y salvaguardar la retención de los clientes y de esta manera también determinar si el almacén cuenta con ese tipo de piezas, partiendo de un catalogo que es la base de datos para este caso, y así generar un algoritmo que permita identificar una imagen por medio de una fotografía.

6. Desarrollo del método

Dentro de los métodos de clasificación de aguacate se distinguen, por encima de los químicos y físicos, aquellos que implican el procesamiento de imágenes, porque no comprometen la integridad del fruto. La simplificación del algoritmo se vuelve el objetivo de los investigadores que deciden desarrollar este tema.

El método para la clasificación del aguacate implementado en este documento se divide en las siguientes partes:

1. Obtención de la imagen.
2. Procesamiento de la imagen para obtener características.
3. Clasificación manual de los ejemplos de entrenamiento.
4. Entrenamiento de la máquina de vectores de soporte.
5. Finalización del método.

6.1. Obtención de la imagen

Se acota el problema de la obtención de la imagen a un ambiente controlado, en el que los parámetros de iluminancia, luminancia, ángulo de la cámara, y distancia de la cámara con respecto al objeto, dejan de ser variables para obtener capturas de las cuales se puedan obtener características que se puedan comparar entre ellas, y por tanto, determinísticas para la clasificación del aguacate.

La construcción del ambiente controlado fue diseñada sobre una caja de madera la cual tiene como medidas 17,5cm de largo, 17,5cm de ancho y 25cm de alto.

En cuya parte superior cuenta con una tapa del mismo material la cual contiene una cinta de luces led de color blanca, alimentada con 12 V provenientes de una fuente para garantizar la tensión constante en las luces y así no tener variaciones en la intensidad dentro del ambiente ya que podría interferir en los datos obtenidos en las capturas de las imágenes y sería un error ya que ese parámetro no sería controlado, también cuenta con un orificio por el cual es ingresada la picamara para la captura de la imagen, y por el otro orificio cuenta con un interruptor que permite encender la cinta de las luces, la

caja en su interior esta forrada por vinilo Contac de color blanco para crear contraste entre el fondo y el aguacate, y de esta manera obtener una imagen con menos parámetros a la hora de hacerle el procesamiento digital a la imagen obtenida del aguacate.

En la figura 7 se muestra las imágenes del ambiente controlado con sus respectivos componentes, los cuales son fundamentales para el buen desarrollo del proyecto.

a) b) c) d)

Figura 7. a) Vista Superior, b) Vista interior iluminada. c) Vista frontal. d) Vista interior sin iluminar.

En la imagen a) se observa el posicionamiento de la Rapsberry Pi, el interruptor, ambas ubicado sobre la tapa superior del ambiente controlado. En la imagen b) se observa el interior del ambiente controlado con la cinta led iluminando su interior, en la imagen c) se observa una vista frontal de lo que es el ambiente controlado, y por último en la parte d se observa el interior del ambiente controlado sin encender la cinta de luces tipo led.

El código en Python implementado para obtener la captura de imagen, procesamiento, y obtención de características se muestra a continuación:

```
from picamera.array import PiRGBArray

from picamera import PiCamera

import time

import cv2

import shutil

import numpy as np

from Tkinter import *


# inicializar la camara

camera = PiCamera()
```

```
camera.resolution = (640, 480)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(640, 480))

# precalentamiento de la camara
time.sleep(0.1)

# captura de video
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        #crear captura cruda
          image = frame.array


          # imagen umbralizada
          blue,g,r = cv2.split(image)
        ret, im_floodfill=cv2.threshold(blue,80,255,cv2.THRESH_BINARY)


          key = cv2.waitKey(1) & 0xFF

        # mascara usada para el flood filling.
        h, w = image.shape[:2]
        mask = np.zeros((h+2, w+2), np.uint8)

        # Floodfill
        filling=cv2.floodFill(im_floodfill, mask, (0,0), 255);
        #cv2.imshow("flood filling", im_floodfill)

        #solo cate
        maskinv=cv2.bitwise_not(im_floodfill)
        maskinv2=maskinv/255
        BinImage = cv2.merge((maskinv2,maskinv2,maskinv2))
        solo=image*BinImage
```

```
        #cv2.imshow("solo cate", solo)


        #BGR promedio
        b,g,r = cv2.split(solo)
        num = np.sum(maskinv2)
        b=np.sum(b)
        g=np.sum(g)
        r=np.sum(r)
        b=b/num
        g=g/num
        r=r/num


        #concatenar imagenes
        flood = cv2.merge((maskinv,maskinv,maskinv))
        contorno = cv2.merge((blue,blue,blue))
        cate2=np.concatenate((image,contorno),axis=1)
        cate3=np.concatenate((flood,solo),axis=1)
        cate4=np.concatenate((cate2,cate3),axis=0)
        cate5 = cv2.resize(cate4,(960,720))
        cate6 = cv2.putText(cate5,"px="+str(num),(480,380),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
        cate7 = cv2.putText(cate6,"b="+str(b),(480,400),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
        cate8 = cv2.putText(cate7,"g="+str(g),(480,420),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
        cate9 = cv2.putText(cate8,"r="+str(r),(480,440),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
        cv2.imshow("Camara", cate9)


          # limpiar la pantalla antes de mostrar una nueva imagen
          rawCapture.truncate(0)
```

```
        if key == ord("q"):
                camera.capture(rawCapture, format="bgr")
            img = rawCapture.array
            #leer registro
            f = open ('registro.txt','r')
            reg = f.read()
            reg = int(reg)
            reg=reg+1
            f.close()
            #escribir registro
            f = open ('registro.txt','w')
            f.write(str(reg))
            f.close()
            f = open('train.txt','a')
            f.write('¥n' + '[' + str(b) + ',' + str(g) + ',' + str(r) + ',' + str(num) + ']' )
            f.close()
            #guardar imagen y mostrar
            cv2.destroyWindow("Frame")
            cv2.imshow("Image", img)
            cv2.waitKey(0)
            cv2.destroyWindow("Image")
            cv2.imwrite("foto-"+str(reg)+".png", img)
            shutil.move("foto-"+str(reg)+".png", "/home/pi/Desktop/base/dia2")
            time.sleep(0.1)
            rawCapture.truncate(0)

    if key == ord("x"):
            break

cv2.destroyAllWindows()
```

6.2. Procesamiento de la imagen para obtener características.

El Procesamiento de imágenes se puede resumir en el algoritmo de la figura 8. Cuando se hace la captura con la Raspnerry Pi, se obtiene una matriz cuadrada en la que cada elemento es una serie de tres números que corresponden a su color, segmentado en los colores primarios de la luz: azul, verde, rojo.

Figura 8. algoritmo del procesamiento de imágenes para obtención de características.

Para el caso de estudio de los aguacates, se ha comprobado experimentalmente que su color característico tiene muy poca intensidad de azul (figura 9), mientras que el fondo (blanco) tiene un porcentaje equitativo entre los colores primarios de luz. Se puede aprovechar esta propiedad para generar un alto contraste entre el fondo del ambiente controlado y el aguacate, y usar este valor como máscara de bordes para el algoritmo floodfill.

Antes de aplicar floodfill, se debe crear la máscara de bordes que necesita como parámetro de entrada para dar límites al relleno de pixeles, esto se hace con el umbral por histéresis, de esta manera de maximiza la característica de alto contraste de la parte azul del aguacate con el fondo del ambiente controlado.

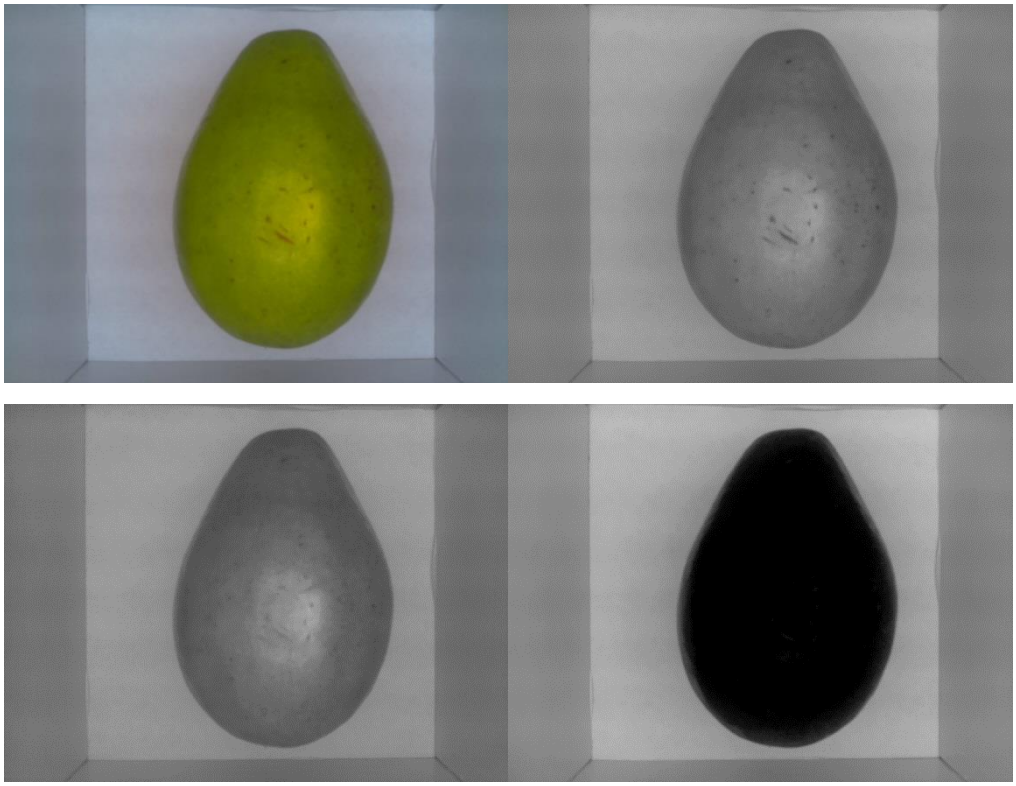


figura 9. Segmentación de la imagen por colores. Superior izquierda: original, superior derecha: verde, inferior derecha: rojo, inferior izquierda: azul.

Lo siguiente es obtener la imagen de solamente el aguacate como preparación para la obtención de características, así que se binariza la imagen en lo útil y no útil para obtener solamente el área de estudio y pasar al siguiente paso (figura 10).

Figura 10. Resultado de la clasificación de pixeles por área de interés.

La característica que se trabaja en este método es el de "color promedio", que consiste el en promedio ponderado aritmético de cada pixel que forma la imagen y pertenece al objeto:

$$Color\ promedio = \left[\frac{\sum px\ azul}{npa}, \frac{\sum px\ verde}{npa}, \frac{\sum px\ rojo}{npa}\right]$$

$$npa = n\acute{u}mero\ de\ p\acute{i}xeles\ del\ aguacate$$

6.3. Clasificación manual de los ejemplos de entrenamiento.

Una vez listo el sistema para la obtención de características por imagen, lo siguiente es hacer un seguimiento del proceso de maduración del aguacate desde una edad temprana (poco maduro) hasta una etapa alta (demasiado maduro), e ir etiquetando los estados de maduración junto con cada característica.

El proceso de maduración del aguacate papelillo desde cosecha oscila entre unos 3 a 5 días, en este caso, se tomaron 4 fotos a cada uno de los 9 ejemplos disponibles durante los 5 días del proceso. El resultado obtenido es una matriz de 180x3 a la que se adjunta una matriz columna de 180 filas con la etiquetas de clasificación para cada uno de los días; estas dos matrices serán los ejemplos de entrenamiento del SVM.

6.4. Entrenamiento de la máquina de vectores de soporte.

Para esta parte se hace uso del algoritmo SVM brindado por la librería de open CV, el proceso es muy sencillo una vez organizada la matriz de características y etiquetas. El código utilizado para esta tarea se muestra a continuación:

```
import cv2 as cv

import numpy as np

import matplotlib.pyplot as plt


caracteristicas=[[3,108,94,],

[6,103,91,],

[5,110,98,],

[5,104,88,],

[5,104,92,],

[7,110,93,],
```

[5,104,87,],

[5,110,92,],

[4,112,91,],

[4,110,91,],

[5,110,92,],

[5,111,92,],

[7,105,89,],

[6,111,94,],

[7,117,101,],

[7,114,98,],

[7,102,84,],

[6,109,94,],

[6,110,95,],

[7,104,85,],

[5,114,98,],

[5,117,105,],

[4,118,111,],

[4,118,107,],

[5,109,90,],

[6,106,90,],

[5,109,96,],

[5,114,97,],

[5,103,87,],

[5,98,83,],

[4,111,98,],

[6,105,89,],

[4,103,86,],

[6,96,76,],

[5,121,105,],

[5,115,98,],

[5,98,79,],

[4,111,89,],

[5,117,100,],

[5,111,99,],

[12,95,110,],

[15,95,116,],

[7,100,118,],

[6,102,119,],

[4,112,113,],

[5,112,119,],

[5,113,116,],

[5,113,111,],

[5,113,111,],

[4,106,105,],

[6,102,110,],

[6,103,111,],

[13,98,115,],

[7,102,119,],

[6,103,113,],

[7,97,117,],

[6,110,118,],

[3,108,116,],

[6,101,113,],

[10,95,108,],

[10,102,117,],

[7,105,115,],

[6,107,114,],

[7,108,114,],

[5,106,103,],

[5,104,104,],

[4,108,103,],

[4,114,108,],

[5,110,112,],

[6,107,108,],

[4,104,103,],

[4,94,96,],

[6,107,115,],

[6,115,117,],

[5,108,116,],

[8,104,117,],

[6,108,119,],

[5,111,121,],

[7,102,109,],

[6,99,111,]]

estados=[[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

```
[1],
[1],
[1],
[1],
[1],
[1],
[1],
[1],
[1],
[1]]

trainData = np.float32(caracteristicas)
responses = np.int_(estados)

svm = cv.ml.SVM_create()
svm.setKernel(cv.ml.SVM_LINEAR)
#POLY polinomial
#RBF funcion de base radial o  gausiana
#SIGMOIG
svm.setType(cv.ml.SVM_C_SVC)
svm.setC(2.67)
svm.setGamma(5.383)
svm.train(trainData, cv.ml.ROW_SAMPLE, responses)
svm.save('svm_lineal.dat')
testData = np.float32(trainData)
result = svm.predict(testData)
print result
```

6.5. Finalización del método

Se puede editar las últimas líneas del algoritmo de captura para agregar en pantalla la clasificación del aguacate en tiempo real, de la siguiente manera:

```
result = svm.predict(np.float32([[b,g,r]]))
if result[1] == 1:
   text='no maduro'
else:
   text='maduro'
cate6 =
cv2.putText(image,"px="+str(num),(10,10),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
cate7 =
cv2.putText(cate6,"b="+str(b),(10,30),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
cate8 =
cv2.putText(cate7,"g="+str(g),(10,50),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
cate9 =
cv2.putText(cate8,"r="+str(r),(10,70),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
cate10=
cv2.putText(cate9,"estado="+text,(10,90),cv2.FONT_HERSHEY_SIMPLEX, 0.5
,(10,16,202), 2 , cv2.LINE_AA)
cv2.imshow("Camara", cate10)


# limpiar la pantalla antes de mostrar una nueva imagen
rawCapture.truncate(0)
```

7. Resultados

La última edición al código permite visualizar, en tiempo real, el estado de maduración actual del aguacate puesto en el ambiente controlado. Se muestra en la figura 11 la distribución de los datos tomados para los estados maduro y no maduro. Se puede

deducir gráficamente que hay una diferencia geométrica importante en la distribución de las dos clases, sin embargo, la cercanía de los datos sugiere la implementación de otro tipo de característica para la determinación de uno o más estados intermedios. En la figura 12se puede observar la salida del programa implementado en Python para aguacates en dos estados de maduración.

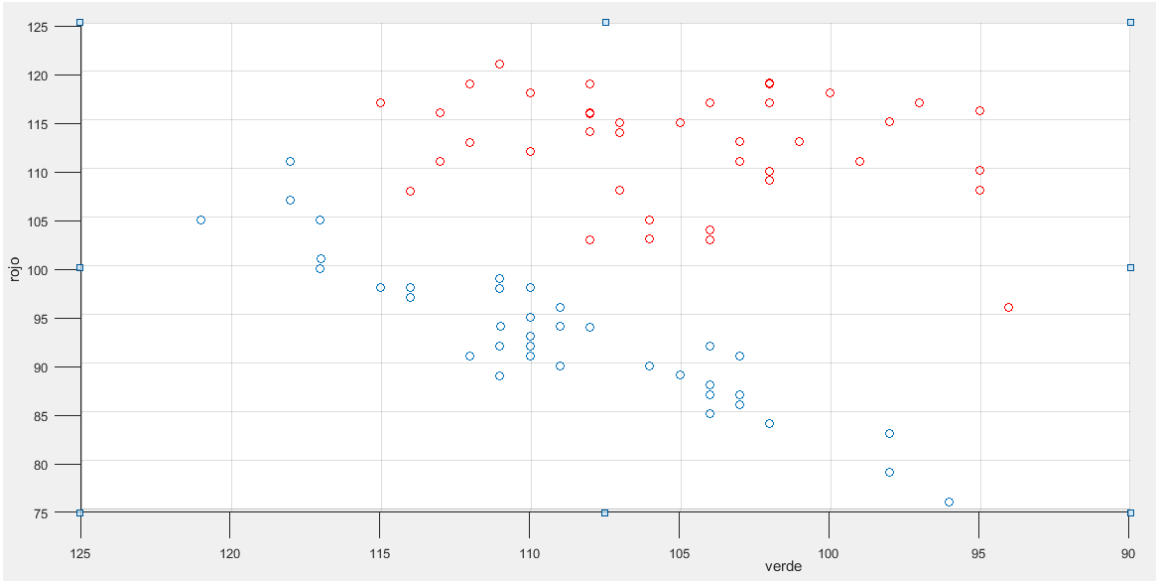


Figura 11. distribución de las características del aguacate para los estados poco maduro (azul) y muy maduro (rojo).

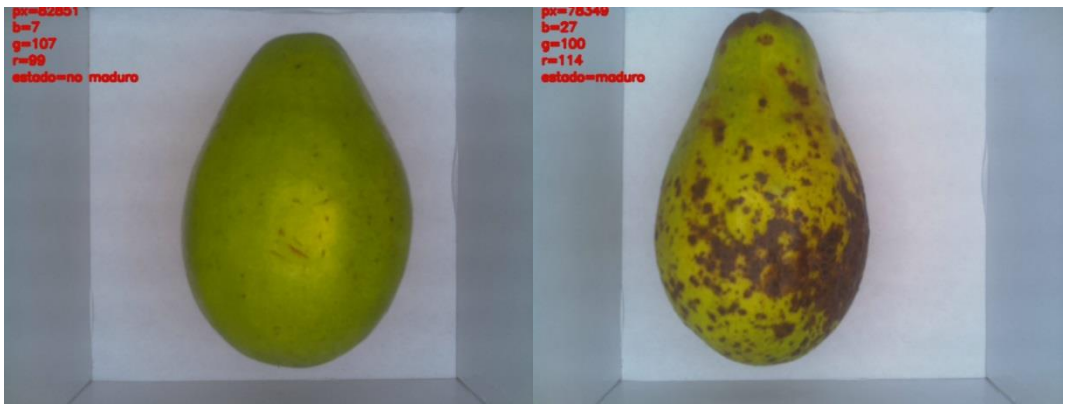


Figura 12. Diferentes estados de maduración desde el algoritmo.

La proximidad en la magnitud de las características es un factor a tomar en cuenta para la adecuada verificación de los parámetros del ambiente controlado, ya que la lectura de los datos tiene una gran sensibilidad a cambios de estos. Un mínimo cambio en el voltaje puede hacer que la intensidad de luz emitida por la fuente varíe un poco, y por

tanto generar lecturas erróneas; un cambio en la exposición de la cámara para tomar una imagen cambia notablemente la nitidez de ella, por lo que los datos entregados por el código puede cambiar importantemente si se utiliza lecturas en video (captura cruda) o si se toma una imagen y a esta por separado se le hace el procesamiento de imágenes.

Además, se debe tener en cuenta que las características serán diferentes si se trata de la especie *papelillo* de aguacate (con la cual se hicieron pruebas en este proyecto), o con otra; por lo que en un proyecto futuro se puede implementar más características de reconocimiento de patrones en imágenes para determinar primero su especie y posteriormente una clasificación por estado de maduración.

.

8. Conclusiones

Se logro desarrollar un algoritmo el cual permite determinar el estado de madures del aguacate (papelillo) empleando el procesamiento digital de imágenes, sobre un ordenador de placa reducida el cual hace nombre a la Rapsberry pi 2B y empleando la cámara Picamara, todo esto sobre un ambiente controlado para garantizar en lo mayor posible la obtención de datos bajo los mismas condiciones como son la intensidad de luz, y también para obtener la imagen sobre las cuales se extrajeron características específicas, para este caso la característica principal es el color, y dependiendo de la variación de este color en cada una de sus estados (poco maduro, maduro, muy maduro) se entrega un resultado de su estado en tiempo real al someter un aguacate al procesamiento digital.

Se logra determinar el estado de madurez con cierto grado de error, ya que no fue posible garantizar que los aguacates que fueron empleados como base de datos perteneciera a una misma cosecha y de un mismo palo, ya que levemente variaban en su esto de madurez, determinando esto por su dureza, y variación de color.

Se desarrollo una interfaz grafica por medio de la cual muestra al usuario el estado de maduración en el que se encuentra el aguacate sometido al análisis en tiempo real.

9. Trabajo futuro

En base a este proyecto realizado se podrá implementar mejoras que permita brindar mayor precisión, una de estas mejoras es incorporar un sensor de presión dentro del ambiente controlado para determinar la dureza del aguacate y así obtener otra característica determinística. Otra consideración para implementar sobre este proyecto sería la de la detección de defectos en el aguacate, para permitir su selectividad partiendo de los resultados obtenidos y así garantizar una mayor calidad de los aguacates al momento de comercializarlos. Se puede trabajar con diferentes tipos de aguacates ya que entre ellos existen demasiadas diferencias, es decir unos varían en color mas que otros, maduran en muchos más días, tienen diferentes formas y diferentes tamaños, también se podrá hacer una comparación entre los resultados obtenidos mediante diferentes métodos de clasificación y determinar cual es el que tiene más exactitud y menos porcentaje de error, y poder de esta manera emplear un método más confiable para este tipo de pruebas. Se tendrá como consideración futura garantizar aguacates que pertenezcan a un mismo lote y el mismo día de cosecha, permitiendo interactuar con los cosechadores y las fincas que lo producen, para tener mas control y confianza en los datos que se obtendrán, y así minimizar el rango de error en lo mayor posible, creando un programa mas confiable y estable a la hora de determinar el estado de maduración en el que se encuentra el aguacate. También es recomendable trabajar sobre el ambiente controlado, optimizándolo en una mayor medida.

Debido a la limitación que presenta la Raspberry Pi 2B en velocidad de procesamiento, y la robustez del programe implementado, se recomienda actualizar el Hardware a uno mas potente como la Raspberry Pi 3B.

10. Bibliografía

[1] Acosta, W. (2009). Evaluación del proceso de rehidratación del liofilizado de aguacate criollo (persea americana mil. Variedad Drimifolya) mediante análisis de imágenes (tesis especialización). Instituto Politécnico Nacional, México DF, México.

[2] Escobar, D. (2016). Diseño e implementación de una herramienta computacional para la identificación del estado de madurez de la granadilla mediante técnicas de visión artificial sobre un ordenador de placa reducida (tesis pregrado). Universidad de Cundinamarca, Fusagasugá, Colombia.

[3] Arias, M, & Sierra, J. (2016). Procesamiento de imágenes para la clasificación de café verde (tesis pregrado). Pontificia Universidad Javeriana, Bogotá DC, Colombia.

[4] Viera, G. (2017). Procesamiento de imágenes usando Opencv aplicado en Rapsberry pi para la clasificación del cacao (tesis pregrado). Universidad de Piura, Piura, Perú.

[5] Gonzales, J. (2017). Reconocimiento de objetos utilizando Opencv y Python en una Rapsberry pi 2 en una tlapalería (tesis pregrado). Universidad Autónoma del estado de México, Texcoco, México.

[6] Barrero, A., Robayo, M. y Jacinto, E. (2015). Algoritmo de navegación a bordo en ambientes controlados a partir de procesamiento de imágenes. Revista Tekhnê, 12(2), 23-34. Universidad distrital Francisco José De Caldas, Manizales, Colombia.

[7] Pencue, E, & León, J. (2014). Detección y clasificación de defectos en frutas mediante el procesamiento digital de imágenes. REVISTA COLOMBIANA DE FISICA, VOL. 35, No.1. 2003. Universidad del Cauca, Popayán, Colombia.

[8] Enrique J Carmona Suarez. (2013). *Tutorial sobre máquina de soporte (SVM)* [PDF file]. Recuperado de http://www.ia.uned.es/~ejcarmona/publicaciones/[2013-Carmona]%20SVM.pdf

[9] Jorge Valverde Rebaza. (2014). *Detección de bordes mediante el algoritmo de canny* [PDF file]. Recuperado de https://www.researchgate.net/publication/267240432_Deteccion_de_bordes_mediante_el_algoritmo_de_Canny