Sistema de reconocimiento de maduración del tomate, mediante el procesamiento de imágenes con OpenCV y Python en una Raspberry Pi 3 B

Chapter · September 2018

DOI: 10.5281/zenodo.2613975

CITATIONS
0

READS
612

3 authors:

Angel Alejandro Rodriguez Aya
National Open University and Distance (Colombia)
6 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Juan Alejandro Chica Garcia
National Open University and Distance (Colombia)
8 PUBLICATIONS **6** CITATIONS
SEE PROFILE

John Alejandro Figueredo Luna
National Open University and Distance (Colombia)
5 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Sistema de reconocimiento de maduración del tomate, mediante el procesamiento de imágenes con OpenCV y Python en una Raspberry Pi 3 B

Ángel A. Rodríguez A.[1]
John A. Figueredo L.[2]
Juan A. Chica G.[3]
[1, 2] Universidad Nacional Abierta y a Distancia
Acacias – Colombia
[3] Corporación Universitaria Autónoma de Nariño
Villavicencio – Colombia

En la actualidad, el procesamiento de imágenes aplicado a diferentes campos de la ingeniería ha tenido un crecimiento elevado, desde el reconocimiento de un simple color para clasificar un objeto, hasta la detección de rostros con el fin de hacer un reconocimiento facial de personas. Hoy en día, gracias a la evolución y el abaratamiento del hardware y el uso de software libre, se puede desarrollar herramientas de bajo costo para aplicaciones de mediana complejidad en el área de procesamientos de imágenes, es así, como en este capítulo se muestra el principio del desarrollo de un sistema de procesamiento de imágenes de bajo costo, basado en una tarjeta Raspberry pi 3 modelo B y el uso de la librería OpenCV en Python para el reconocimiento del estado de maduración del tomate, en aras de realizar una clasificación por tamaño y color del fruto, con el fin de ser aplicado a futuro en la generación de valor agregado para los productores y comerciantes de tomate en el departamento del Meta. Como resultado se obtiene el prototipo piloto capaz de reconocer la maduración de diferentes tomates, aplicando el procesamiento de imágenes con la librería OpenCV mediante la técnica de segmentación de imágenes realizando correcciones en la imagen a procesar, minimizar el ruido y las perturbaciones.

1. INTRODUCCIÓN

El procesamiento digital de imágenes es el principio de la visión por computador, en el cual por medio de una imagen digital (foto, imagen o fotograma) se puede analizar diferentes características en ella, tales como color, forma, textura, tamaño, entre otras, para cumplir una finalidad dentro de un algoritmo ejecutado en un computador o hardware con similares funcionalidades [1], es así como en los últimos años se ha desarrollado software y librerías específicas para esta función, y así poder realizar algún tipo de proceso encaminado hacia la visión por computador en la industria, robótica y automatización de procesos [18]. En el área de la ingeniería, se pueden encontrar diferentes softwares que pueden llevar a cabo esta funcionalidad, como lo es el caso de MATLAB o LABVIEW, estos softwares son aplicados a diferentes campos de la ingeniería, pero no son desarrollados específicamente al procesamiento de imágenes, aunque parte de ellos dedican diferentes Toolbox y herramientas dedicadas a esta área [8]; con el fin de realizar el procesamiento digital de imágenes, cabe señalar que los dos softwares mencionados anteriormente son licenciados y el costo de uso y/o adquisición es bastante elevado para aplicaciones de baja escala o dirigidos a pequeños proyectos, donde el presupuesto juega un papel importante a la hora de llevarlos a cabo.

Por otra parte, también existen otras alternativas de software para realizar el procesamiento digital de imágenes digitales y al alcance de cualquier investigador. Este es el caso de la librería OpenCV de Intel, bajo licencia BSD, siendo gratuita para uso académico y comercial [11], la cual puede ser utilizada con diferentes lenguajes de programación, tales como C++, Python y Java, además, puede ser instalada en diferentes sistemas operativos como por ejemplo Windows, MAC OS, iOS, Android, Linux y Raspbian; este último sistema operativo soportado para las tarjetas Raspberry Pi [21].

2. HARDWARE Y SOFTWARE PARA EL PROCESAMIENTO DIGITAL DE IMÁGENES

En el mercado existe una gran variedad tanto de hardware y software que puede ser utilizado para el procesamiento digital de imágenes, los cuales pueden variar en la capacidad de procesamiento, robustez, velocidad y costo, esta última ítem es uno de los más importantes criterios a la hora de su implementación; ya que presenta los mayores limitantes a la hora de seleccionar el conjunto de herramientas (Hardware y software) para llevar a cabo el proyecto. Esta investigación se basa principalmente, en el desarrollo de una herramienta de bajo costo de implementación para el procesamiento digital de imágenes en la determinación de estado de maduración del tomate (maduro o verde) y tamaño, con el fin de clasificar automáticamente estos dos criterios bajo ambientes controlados dentro de una cadena de producción. De acuerdo a lo anterior se determinó las herramientas de hardware y software a implementar [23].

2.1 Hardware

El hardware para el procesamiento digital de imágenes, se basó en la relación costo-beneficio para la finalidad de la investigación, el cual será el encargado de proveer un procesamiento eficaz a un bajo costo, además, realizar la función

[1] *angel.rodriguez@unad.edu.co*
[2] *john.figueredo@unad.edu.co*
[3] *juan.chica@aunarvillavicencio.edu.co*

de procesar la información en tiempo real y sea capaz de enviar las señales a los actuadores con el fin de clasificar los tomates de acuerdo a los criterios antes mencionados [17]. El hardware que mejor se amolda a los requerimientos del sistema es la tarjeta Raspberry Pi 3 modelo B. La cual la define según la Raspberry PI Foundation como un computador de bajo costo y de alto rendimiento, desarrollada con el criterio de que cualquier persona pueda acceder a este hardware y pueda aprender, resolver problemas y divertirse [15]. La Raspberry Pi Modelo B (Figura 1), cuenta con un sistema operativo de distribución libre basado en Debian llamado Raspbian, por otra parte, esta tarjeta tiene un costo aproximado de $40 Dólares y cuenta con las siguientes características:

- Chipset Broadcom BCM2837 a 1,2 GHz
- ARM Cortex-A53 de 64 bits y cuatro núcleos
- LAN inalámbrica 802.11 b/g/n
- Bluetooth 4.1 (Classic y Low Energy)
- Coprocesador multimedia de doble núcleo Videocore IV
- Memoria LPDDR2 de 1 GB
- Compatible con todas las últimas distribuciones de ARM GNU/Linux y Windows 10 IoT
- Conector micro USB para fuente de alimentación de 2,5 A
- 1 puerto Ethernet 10/100
- 1 conector de vídeo/audio HDMI
- 1 conector de vídeo/audio RCA
- 1 conector de cámara CSI
- 4 x puertos USB 2.0
- 40 pines GPIO
- Antena de chip
- Conector de pantalla DSI
- Ranura de tarjeta microSD
- Dimensiones: 85 x 56 x 17 mm

Estas características la postulan como una herramienta idónea con gran potencial para desarrollar el procesamiento digital de imágenes en la selección de los tomates por tamaño y tipo de maduración.

Figura 1. Tarjeta Raspberry Pi 3 modelo B (https://www.raspberrypi.org/)

Para la captura de las imágenes, ya sea en video o fotos, se implementó la cámara Raspicam (Figura 2), desarrollado para conectarse directamente en uno de los puertos dedicados CSi para obtener las imágenes a procesar en la Raspberry. Esta cámara cuenta con 5 megapíxeles con un lente de foco fijo y es capaz de tomar imágenes de 2592 x 1944, siendo compatible con el formato de video 1080p30, 720p60 y 640x480p60/90, todas estas estas características mencionadas anteriormente por un costo aproximado de $8 Dólares.

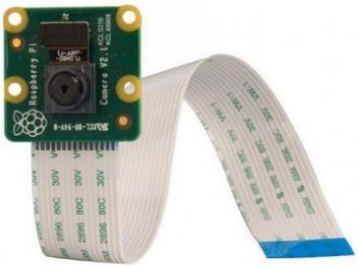

Figura 2. Cámara Raspicam (https://www.raspberrypi.org/)

2.2 Software

El sistema operativo que se implementará en la tarjeta Raspberry Pi 3 modelo B, será el nativo de la propia tarjeta, el cual es Raspbian, este sistema operativo se basa en Debian y es el sistema operativo oficial para la tarjeta, por otro

lado, cabe resaltar que existen varios sistemas operativos para la tarjeta Raspberry, tales como Windows IoT, Ubuntu Mate, entre otros, lamentablemente la mayoría de ellos ralentiza el funcionamiento de la tarjeta [8]. Por otra parte, el lenguaje que se trabajará para el desarrollo del algoritmo que cumpla la función del procesamiento de imágenes será Python, el cual ya viene instalado dentro del sistema operativo Raspbian. Python es un lenguaje de programación interpretado, el cual su filosofía se basa en una sintaxis que favorezca el código legible [2], además, otra de las características de este lenguaje de programación es que es multiparadigma, ya que soporta programación orientada a objetos, programación imperativa y programación funcional, siendo este multiplataforma lo que lo hace ideal en una gran variedad de aplicaciones.

La librería principal adicional para el procesamiento de imágenes es OpenCV, la cual es desarrollada por Intel para el procesamiento digital de imágenes, esta librería puede aprovechar el procesamiento de varios núcleos, además, puede aprovechar la aceleración de hardware de la plataforma informática heterogénea subyacente [11]. Esta librería se complementa con la librería NumPy para Python, la cual agrega soporte para vectores y matrices, haciéndola ideal para el procesamiento de imágenes con el fin de optimizar las operaciones matriciales necesarias cuando es necesario segmentar la información obtenida en una imagen.

3. TÉCNICAS Y MÉTODOS APLICADOS EN OPENCV PARA EL PROCESAMIENTO DIGITAL DE IMÁGENES

Existen diferentes técnicas para extraer información de una imagen digital, las cuales pueden ser, desde la umbralización del color de un pixel para el reconocimiento de un color o gamas de colores de un objeto, hasta técnicas más avanzadas como la correlación digital de imágenes enfocada a encontrar patrones de objetos para su reconocimiento. Para este caso se utilizará la primera técnica mencionada con diferentes mascaras para determinar la gama de colores del objeto y su área [2].

3.1 Conversión de Colores BGR a HSV

En los sistemas computacionales existen varios modelos para representar un color, uno de los más sencillos y utilizados por los sistemas computacionales son el modelo de representación en RGB o BGR, el cual se basa en los tres colores principales (Rojo, Verde, Azul) basados en la intensidad o nivel en un pixel de alguno de estos colores, o en la combinación de ellos para determinar un color derivado de los colores RGB, como por ejemplo, el color rojo se representa del siguiente modo (255,0,0), para un sistema (RGB) con niveles de 0 a 255 (8 bits), en donde se tiene un nivel máximo del rojo que equivale a un 255 y no hay presencia del color verde ni azul. Ahora, el caso para representar un color verde y azul respectivamente se tendría las siguientes codificaciones (0,255,0) y (0,0,255), lo que permite en la combinación de las intensidades de estos tres colores para generar toda la gama de colores disponibles y que se puedan representan según la cantidad de bits [23]. Por otra parte, existe otro modelo más utilizado en el procesamiento digital de imágenes, el cual es el modelo HSV, en cual se representa un color por medio de su Matiz, Saturación y Luminosidad, este modelo tiene la particularidad de que su codificación de color se encuentra definido solo por la variable Matiz y tanto la saturación como la luminosidad son factores de que tan intenso es el color y lo oscuro o claro que puede ser el color a determinar, por lo que lo hace uno de los modelos más apropiados para la umbralización de un pixel dentro de una imagen en cuanto a la característica de color (Figura 3).

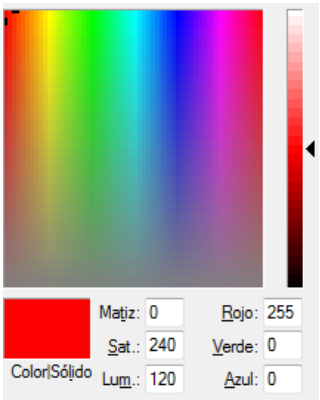

Figura 2. Ejemplo de codificación del color Rojo los Modelos HSV y BGR

3.2 Establecimiento de rango de colores del objeto

Para el reconocimiento del color de un objeto, se hace necesario delimitar los límites de los rangos de color que contiene el objeto a encontrar dentro de la imagen, para ello se hace necesario evaluar los rangos y tonalidades de color que pueda tener el objeto con diferentes tipos de luz, lo anterior, con el fin de tener un rango de colores que pueda contener la mayor cantidad de pixeles del objeto dentro de la imagen, en aras de obtener una imagen binarizada y así ser procesada con mayor facilidad. Una imagen binarizada se compone por una imagen que solo contiene dos colores en cada uno de los pixeles, normalmente definidos por un 1 y un 0 o blanco y negro, con la finalidad de tener solo dos factores de decisión en cada pixel de la imagen para su posterior procesamiento, lo que se entiende en procesamiento de imágenes como mascara (Figura 4).

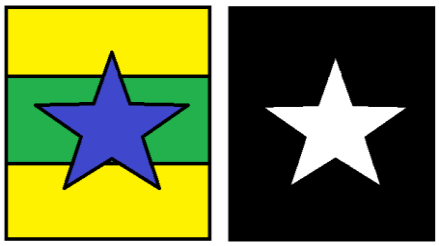

Figura 3. Ejemplo de una imagen binarizada para el color Azul

3.3 Transformaciones morfológicas en una imagen

Las transformaciones morfológicas en una imagen tienen un papel importante a la hora de hacer un procesamiento digital de imágenes, puesto que algunas veces al capturar una imagen, estas no se encuentran en condiciones ideales de luz, foco y en algunos casos presencia de ruido en la imagen, por esta razón la librería OpenCV dispone de diversas transformaciones morfológicas que pueden ayudar en algunos casos a mejorar la imagen con el fin de obtener una nueva imagen más idónea para su proceso, algunas de ellas son: Erosion, Dilation, Opening, Closing, Morphological Gradient, Top Hat, Black Hat. A continuación, se presenta un ejemplo de la transformación morfológica Opening [2], la cual puede ayudar a eliminar el ruido de la imagen (Figura 5).

Figura 5. Transformación morfológica Opening sobre una imagen binarizada
(https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html)

3.4 Contornos sobre una imagen

Los contornos sobre una imagen son una curva que encierra todos los puntos continuos del mismo color, esta técnica es una herramienta útil para el análisis de formas y la detección y el reconocimiento de objetos, la instrucción de contornos en OpenCV se debe utilizar bajo imágenes binarizadas, con el fin de agrupar los conjuntos de pixeles del mismo tipo, esta técnica se utiliza como punto de partida para encontrar el contorno del objeto y su ubicación dentro del fotograma, además, permite calcular el área del contorno encontrado, generando un área aproximada del objeto a buscar (Figura 6).

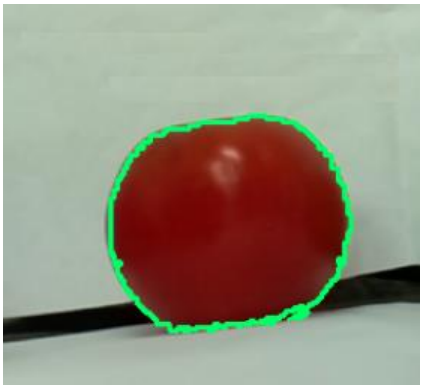

Figura 4. Contorno con OpenCV sobre una máscara de color rojo

4. DISEÑO DEL ALGORITMO PARA LA DETECCIÓN DE TOMATES POR TAMAÑO Y MADURACIÓN

El desarrollo del algoritmo implementado en Python con la librería OpenCV se diseñó bajo diferentes etapas o pasos para llevar a cabo el reconocimiento del fruto del tomate por tamaño y maduración, para ello, fue necesario aplicar las diferentes técnicas mencionadas anteriormente. A continuación, se muestran los pasos que se ejecutaron para su desarrollo.

4.1 Captura de la imagen en un ambiente controlado

La adquisición de la imagen del objeto a procesar se basó en la captura de fotogramas por medio de la Raspicam y la tarjeta Raspberry Pi 3 modelo B en un ambiente controlado, para ello fue necesario implementar un pequeño estudio (Figura 7) para tratar de mantener las condiciones de luz del ambiente constante y un fondo de un solo color, además, se hizo necesario tener un punto de referencia del objeto buscado y la distancia de la cámara, lo anterior para tener un punto fijo en cada medición del tomate y así poder determinar un área aproximada con menor porcentaje de error.

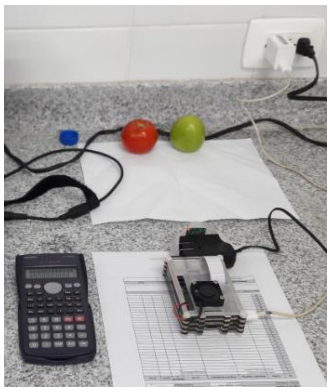

Figura 5. Captura de la imagen con la Raspicam

4.2 Calibración de las tonalidades de maduración en los tomates

La imagen capturada consiste en una trama de 15 fotogramas, los cuales como primera medida se hace necesario la conversión de BGR a HSV para poder desarrollar un mejor proceso de análisis de la imagen, como proceso intermedio antes de la binarización de la imagen, se hizo necesario realizar una calibración de cada una de las máscaras para reconocer dos tipos de estados, maduro y verde, con el fin de tener el ancho de espectro de colores correspondientes a cada uno de estos estados, para llevar a cabo esta función, se tomaron como referencia 50 tomates maduros en diferentes puntos de su etapa de maduración y 50 tomates verdes, en aras de tener una amplia gama de tonalidades y hacer una máscara de acuerdo a los criterios de maduración, por otra parte, se realizó un algoritmo adicional para realizar la calibración de las máscaras para así tener la tabla patrón de los rangos de medición (Figura 8).

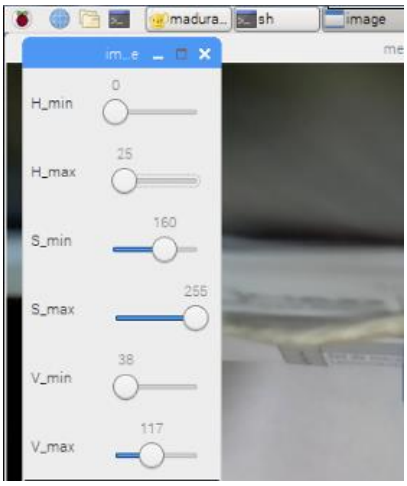

Figura 6. Barras de calibración para la detección de niveles HSV para los tonos de color de maduración de tomates

La calibración de la gama de colores correspondiente al punto de maduración (Tonalidades rojas), fue necesario realizar dos mascaras para este tono de color, puesto que las gamas de colores del rojo se encuentran a los dos extremos del espectro de tonalidades en HSV. Como proceso final se hace necesario realizar la suma de las dos mascaras resultantes del color rojo y así obtener una única mascara para esta tonalidad (Figura 9).

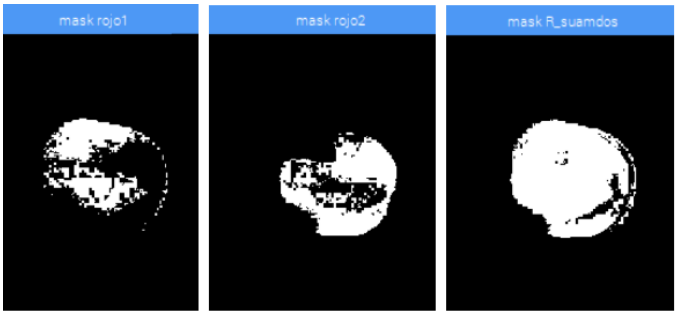

Figura 9. Mascara Rojo 1, Mascara Rojo 2 y máscaras rojas sumadas

4.3 Tabla de valores promedio de los niveles HSV para el estado de maduración del tomate

Al realizar el tamizaje de los 50 tomates maduros y 50 tomates verdes en diferentes etapas de su maduración, se logró determinar los límites de cada una de las máscaras para el reconocimiento de cada uno de los dos estados del tomate objeto de esta investigación, permitiendo tener un patrón de colores como base para el algoritmo de reconocimiento de la maduración de tomates. A continuación, en la Tabla 1 se muestran de valores obtenidos al realizar el tamizaje.

Tabla 13. Límites para los niveles HSV para cada mascara

Estado de Maduración	Mascara	Limites	Nivel H	Nivel S	Nivel V
Tomate Maduro	Mascara rojo 1	Límite inferior	0	65	75
		Límite superior	12	255	255
	Mascara rojo 2	Límite inferior	116	193	4
		Límite superior	255	255	246
Tomate Verde	Mascara verde	Límite inferior	20	77	84
		Límite superior	75	255	205

4.4 Aplicación de transformaciones morfológicas

En la siguiente etapa del algoritmo se realizó la aplicación de diferentes transformaciones morfológicas para mejora la calidad de la imagen en cuanto a textura y forma, con el fin de llevar a cada una de las máscaras a una aproximación cercana a la imagen real, en este apartado se realizaron las transformaciones morfológicas en el siguiente orden: Closing, para completar algunos pixeles no reconocidos dentro de la máscara, seguido de Dilation con el fin de seguir completando los puntos no reconocidos en cada mascara cuando se realiza la umbralización por color. Después de realizar dichas transformaciones, se obtuvo una aproximación de 83% de la imagen real sin aplicar ninguna transformación, hasta llegar al 94% de aproximación (Figura 10) después de aplicar las transformaciones morfológicas.

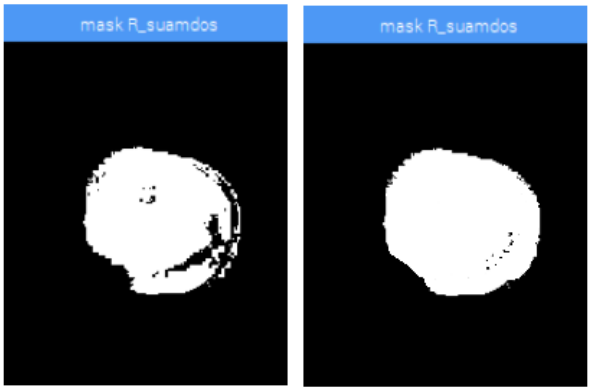

Figura 10. Mascara antes y después de aplicar las transformaciones morfológicas

4.5 Determinación del contorno del objeto

El algoritmo a este punto ya tiene una aproximación clara y definida del objeto que desea reconocer, pero hasta este momento no se ha determinado el contorno que encierra cada una de las máscaras. Para ello se utiliza la función cv2.findContours de OpenCV (Figura 11) a cada mascara para encontrar todos los contornos, posteriormente, es necesario hacer una comparación de todos los contornos encontrados de la misma masca y delimitarlos por tamaño, lo anterior con el fin de eliminar porciones de contorno por fuera de la imagen y solo seleccionar el contorno que tenga el mayor tamaño, esta técnica es utilizada para determinar un único contorno para cada mascara, además, permite tener una visión clara de ella, lo que permite imprimir los contornos en la imagen original.

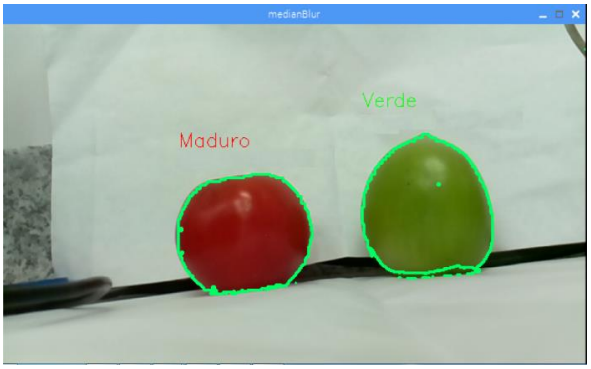

Figura 11. Contornos de las máscaras maduro y verde

4.6 Determinación del área aproximada del tomate

Para la determinación del área aproximada de cada tomate se hace necesario tener como referencia la distancia del tomate a reconocer y la Raspicam, para este caso todas las mediciones del área se hicieron a 25 cm de distancia, para realizar un cálculo aproximado del área, para ello se hicieron cortes transversales de 20 tomates y se hizo su respectiva medición hallando un área aproximada de manera teórica para luego compararla con el tamaño de cada mascara, así se puede obtener una formula aproximada entre ambas mediciones, adicionalmente, se implementó el comando Contour Approximation de OpenCV como punto de comparación, obteniendo un cálculo del área de la sección trasversal aproximada con un error menor al 4,13%(Figura 12).

Figura 72. Calculo del área aproximada del tomate

5. CONCLUSIONES

Los resultados obtenidos en este punto de la investigación denotan la factibilidad de usar el sistema de bajo costo, basado en una tarjeta Raspberry Pi 3 Modelo B con el sistema operativo Raspbian y el uso de la librería OpenCV, en aras de realizar el reconocimiento de imágenes dentro de un ambiente controlado, lo anterior debido a que en la actualidad se presentan tiempos de aproximadamente 1,1 segundos en el procesamiento completo del algoritmo. Por otra parte, cuando se realizan procesos bajo ambientes no controlados, se hace necesario aplicar una mayor cantidad de transformaciones morfológicas y otras funciones que posee OpenCV, lo que incrementa el tiempo de procesamiento del sistema.

Los resultados obtenidos del área de la sección transversal de los tomates son satisfactorias, puesto que se tiene un error cercano al 4% respecto del tomate real y cada una de las máscaras, en este momento, vale la pena aclarar que esta seguirá siendo un área aproximada, ya que para encontrar un volumen o una masa aproximada del tomate será necesario como mínimo tomar dos imágenes del tomate con cámaras ubicadas a 90 grados una de la otra, esto se debe realizar debido a que la contextura del tomate no es totalmente esférica, con la implementación de esta configuración se espera minimizar el error y mejorar la selección del tomate por un tamaño adecuado.

Por otra parte, este trabajo no ha culminado actualmente, lo que se ha documentado hasta el momento se centró en el reconocimiento y procesamiento de imágenes, parte del sistema que tiene la mayor complejidad y dedicación en esta investigación, además, el reconocimiento del fruto en este caso será una condición para generar una señal que se emitirá por los puertos GPIO de la Raspberry al sistema de control y potencia que se desarrollará en una tarjeta Arduino.

Finalmente, el procesamiento digital de imágenes es una técnica que consume una gran cantidad de hardware. Para esta investigación y el algoritmo implementado, se encontró que la carga del procesador en la Raspberry Pi 3 fue cercana al 48%, lo que se espera que para un procesamiento de imágenes más robusto no sería factible la implementación de una Raspberry Pi para llevar tareas que necesiten procesamiento en tiempo real, para ello ya se recomendaría utilizar un computador embebido con mejores prestaciones, como por ejemplo uno con procesador Intel I5 o I7, para obtener mejores resultados. Por otra parte, cabe resaltar que existen otras alternativas para mejorar el rendimiento de una Raspberry Pi, la cual es conectarle un disco duro ya sea mecánico o de estado sólido, lo que podría llegar a aumentar la velocidad de procesamiento hasta 2.5 veces más rápido en comparación con la tarjeta microSD en donde se encuentra actualmente el sistema operativo de la Raspberry, lo anterior debido a que la velocidad de transferencia de datos de una microSD Clase 10, es más baja que la de un disco duro conectada por un puerto USB 2.0.

REFERENCIAS

[1] Acero, Á., Cano, A. & Builes, J. (2015). Sistema de clasificación por visión artificial de mangos tipo Tommy. Revista UIS Ingenierías 14(1), 21-31.
[2] Mordvintsev, A. & Abid, K. (2017). OpenCV-Python Tutorials. Online [Feb 2018].
[3] Bird, S., Klein, E. & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media.
[4] Elfring, J. (2013). Image Processing Using OpenCV. Online [Feb 2018].
[5] García, G. et al. (2015). Learning image processing with opencv. Packt Publishing Ltd.
[6] Kochláň, M. et al. (2014). WSN for traffic monitoring using Raspberry Pi board. In Federated Conference on Computer Science and Information Systems.
[7] Komarudin, A., Teguh, A. & Atmadja, W. (2015). Designing License Plate Identification through Digital Images with OpenCV. Procedia Computer Science 59, 468-472.
[8] Kirby, J. Chapman, L. & Chapman, V. (2018). Assessing the Raspberry Pi as a low-cost alternative for acquisition of near infrared hemispherical digital imagery. Agricultural and Forest Meteorology 259, 232-239.
[9] Lander, G. et al. (2009). Appion: an integrated, database-driven pipeline to facilitate EM image processing. Journal of structural biology 166(1), 95-102.
[10] Mordvintsev, A. & Abid, K. (2017). OpenCV-Python Tutorials. Online [Mar 2018].
[11] OpenCV (2018). OpenCV. Online [Feb 2018].
[12] Osroosh, Y., Khot, L. & Peters, T. (2018). Economical thermal-RGB imaging system for monitoring agricultural crops. Computers and Electronics in Agriculture 147, 34-43.
[13] Parker, J. (2010). Algorithms for image processing and computer vision. John Wiley & Sons.
[14] Qureshi, B. et al. (2016). Performance of a Low Cost Hadoop Cluster for Image Analysis in Cloud Robotics Environment. Procedia Computer Science 82, 90-98.
[15] Raspberry PI Foundation. (2018). Raspberry PI. Raspberry PI Foundation. Online [Mar 2018].
[16] Richardson, M. & Wallace, S. (2012). Getting started with raspberry PI. O'Reilly Media, Inc.
[17] Rodríguez, A., Figueredo, J. & Chica, J. (2018). Economic and environmental benefits in the production of photovoltaic solar electric energy using a solar tracking system in the municipality of Acacias - Meta (Colombia). International Journal of Engineering and Technology 10(2), 345-352.
[18] Santacoloma, G., Cifuentes, J. & Sánchez, L. (2007). Selección de características orientada a sistemas de reconocimiento de granos maduros de café. Scientia et technica 3(35), 139-144.
[19] Seidenari, L. & Lisanti, G. (2010). Image Processing with OpenCV. Online [Feb 2018].
[20] Sugano, H. & Miyamoto, R. (2010). Highly optimized implementation of OpenCV for the Cell Broadband Engine. Computer Vision and Image Understanding 114(11), 1273-1281.
[21] Umesh, P. (2012). Image Processing in Python. CSI Communications.
[22] Van der Walt, S. et al. (2014). Scikit-image: Image processing in Python. Peerj 2, e453.
[23] Yan, Y. & Huang, L. (2014). Large-scale image processing research cloud. In The Fifth International Conference on Cloud Computing, GRIDs, and Virtualization.
[24] Yu, Q. et al. (2004). Ch OpenCV for interactive open architecture computer vision. Advances in Engineering Software 35(8–9), 527-536.