Artículo de Investigación. Revista Killkana Técnica. Vol. 1, No. 2, pp. 21-30, Mayo-Agosto, 2017.
p-ISSN 2528-8024 / e-ISSN 2588-0888. Universidad Católica de Cuenca

Clasificador de imágenes de frutas basado en inteligencia artificial

Fruit image classifier based on artificial intelligence

Diego Heras
Universidad Católica de Cuenca
Cuenca, 010150, Ecuador
dherasb@ucacue.edu.ec

Resumen

Las aplicaciones de visión artificial y el análisis de imágenes son capaces de resolver varios problemas del sector industrial, científico o de seguridad en la actualidad. La clasificación de imágenes es muy útil en la automatización de procesos en una empresa. Para realizar una tarea de clasificación de imágenes se requiere hacer la extracción de características que identifiquen a cada tipo de imagen como por ejemplo: color, forma, textura. En el presente trabajo se requiere implementan los algoritmos para la construcción de un clasificador de imágenes de frutas basado en la extracción de las características del color de las imágenes en determinadas regiones de interés. Para el desarrollo del clasificador de imágenes de frutas se utiliza la técnica de extracción del histograma a color en tres dimensiones y con la implementación de algoritmos de inteligencia artificial se efectúa la clasificación automática de imágenes. El conjunto de datos utilizado consiste en: cuatro clases de frutas con el número variable de imágenes por cada clase, luego se preparan las imágenes seleccionando las regiones de interés mediante técnicas de enmascaramiento y se las divide en dos grupos de datos: Los datos de entrenamiento y los datos de prueba. Luego de entrenado el clasificador, se realizan pruebas de clasificación para evaluar la eficacia del clasificador de imágenes de frutas. Esta metodología de construcción e implementación del clasificador se puede usar en varias aplicaciones según las clases de imágenes de objetos a analizar en casos similares y automatizar procesos de clasificación y reconocimiento de objetos.

Palabras clave: Color, clasificador, histograma, inteligencia artificial, imágenes, python, visión artificial.

Abstract

Machine vision and image analysis applications are nowadays capable of solving various problems in the industrial, scientific or safety sectors. Image classification is very useful for the automation of processes in a company. In order to perform an image classification task, the features identifying each kind of image, such as color, shape, and texture, must be extracted. In the present work, it is necessary to implement the algorithms for the construction of a fruit image classifier, based on image color features extraction in certain regions of interest. For developing the fruit image classifier, the three-dimensional color histogram extraction technique is used, and with the implementation of artificial intelligence algorithms, image automatic classification is accomplished. The dataset used consists of: four fruit types with a varied number of images per class, then the images are prepared by selecting regions of interest through the use of masking techniques and they are then divided into two datasets: training data and test data. After the classifier is trained, classification tests are performed to evaluate the effectiveness of the fruit image classifier. This classifier implementation and construction methodology can be used in various applications, depending on the types of object images to be analyzed in similar conditions, and to automate classification and object recognition processes.

Key words: Color, classifier, histogram, artificial intelligence, images, python, artificial vision.

I. Introducción

En la actualidad los sistemas de visión artificial se emplean cada vez más en la industria agroalimentaria en procesos de inspecciones de calidad o clasificación, ya que permiten automatizar las prácticas manuales y estandarizan técnicas y eliminan costosas tareas humanas de inspección. La mayor ventaja es la objetividad y consistencia en largos periodos de tiempo ya que la objetividad de la visión humana sufre de limitaciones en la percepción visual [1].

Los sistemas de visión por computador son potentes herramientas para la inspección automática de frutas y verduras. Los objetivos de las aplicaciones típicas de tales sistemas incluyen la clasificación, estimación de la calidad según características externas o internas, el seguimiento de los procesos de la fruta durante el almacenamiento o la evaluación de los tratamientos experimentales. Las capacidades de un sistema de visión por computador van mas allá de la limitada capacidad humana, ya que permiten evaluar a largo plazo los procesos de forma objetiva [2].

En los procesos de clasificación manual existe un riesgo relativamente elevado de error humano, ya que las decisiones tomadas por los operarios se ven afectadas por factores psicológicos tales como la fatiga o los hábitos

adquiridos. Un estudio llevado a cabo con diferentes variedades de manzanas, donde personal cualificado comparó varios parámetros de forma, tamaño y color, mostró la limitada capacidad humana para reproducir la estimación de la calidad, lo que los autores definen como "inconsistencia" (Miller, 1991; Paulus et al., 1997). La visión por computador está simplificando estos tediosos y subjetivos procesos de control de calidad en la industria [2].

Por otra parte la extracción de características o información de imágenes se puede realizar por diversos métodos como: textura, tamaño, color, entre otras.

Se implementa un clasificador de imágenes basado en la extracción del "histograma RGB" en tres dimensiones como característica que identifica la clase de imagen. Luego de extraída esta información proveniente de un banco de imágenes previamente digitalizadas, se entrena a una máquina de aprendizaje supervisado para una posterior evaluación en el reconocimiento y clasificación de las imágenes de frutas.

El objetivo del presente trabajo es el de implementar un clasificador de imágenes de frutas basado en algoritmos de inteligencia artificial que permita clasificar imágenes de frutas en función de los colores característicos como parámetros únicos de identificación y clasificación de imágenes.

II. Fundamentos esenciales para la construcción del clasificador de imágenes de frutas

La finalidad de la visión artificial es la de extraer información del mundo físico mediante imágenes y con el uso de un computador, se trata de cuantificar detalles del mundo real como: el brillo, el color, la forma, que pueden provenir de imágenes estáticas, tridimensionales o de vídeo [3].

A lo largo de los años se han estudiado muchos fundamentos que han dado forma a la ciencia de la visión artificial y gracias al avance de nuevos computadores con mayor capacidad de procesamiento se pueden realizar operaciones matriciales de imágenes en tiempo real.

A. *Digitalización de imágenes*

Las imágenes son digitalizadas y representadas como matrices dimensionales donde cada elemento de la matriz corresponde a un valor numérico específico. A cada elemento de la matriz se le denomina "Pixel" y es la unidad más pequeña que compone una imagen.

Adquirir imágenes digitales es pasar la información de un objeto tridimensional del mundo real a una imagen bidimensional, desde un espacio continuo a un espacio discreto de información, esto se lleva acabo mediante una lente que adquiere la imagen y la plasma en un dispositivo digital sensible a la luz como un dispositivo de carga acoplada (CCD). Luego esta información es muestreada y cuantizada.

La imagen digitalizada y cuantizada es representada en forma de una matriz numérica donde cada pixel puede ser ubicado por un par coordenado $I(x, y)$.

$$I(x,y) = \begin{bmatrix} 2 & 4 & 3 \\ 3 & 4 & 5 \\ 6 & 2 & 3 \end{bmatrix} \qquad R(x,y) = \begin{bmatrix} 3 & 4 & 3 \\ 2 & 3 & 5 \\ 6 & 7 & 3 \end{bmatrix}$$

$$G(x,y) = \begin{bmatrix} 7 & 5 & 3 \\ 3 & 4 & 5 \\ 5 & 3 & 4 \end{bmatrix}$$

$$B(x,y) = \begin{bmatrix} 3 & 4 & 4 \\ 4 & 5 & 5 \\ 4 & 2 & 7 \end{bmatrix}$$

FIG. 1. Representación matricial de una imagen en escala de grises y de una imagen a color.

Según la información del brillo contenido en cada pixel de una imagen se la puede clasificar como:

1) Imágenes bitonales:
   Compuestas de dos colores, negro con un valor de 0 y blanco con un valor de 255
2) Imágenes en escala de grises:
   Compuestas de una gama de 256 niveles de grises en su representación
3) Imágenes a color:
   Compuestas de tres matrices monocromáticas con 256 niveles de representación: Rojo, verde y azul, es decir de sus siglas en inglés (RGB) [3].

En la Fig. 1, se puede apreciar una imagen monocromática y otra imagen a color definida por tres matrices monocromáticas en los colores: rojo (R), verde (G) y azul (B).

La Fig.2 muestra el proceso de captura y digitalización de una imagen y sus posibles representaciones.

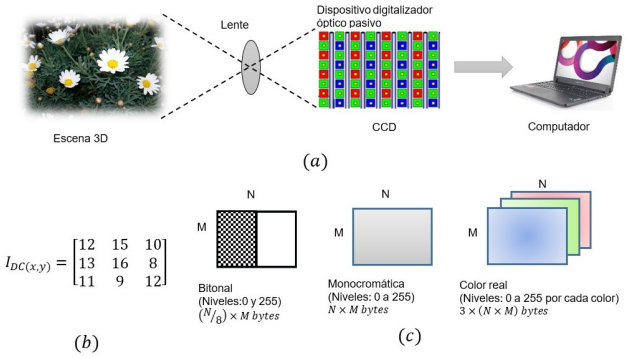

FIG. 2. a) Captura de una imagen con un dispositivo pasivo digital. En b) imagen muestreada en formato de una matriz (MxN). En c) los tres tipos de imágenes digitales con su tamaño en bytes.

La adquisición de la imagen es la parte más importante de la clasificación, ya que representa la materia prima para el procesamiento de la imagen que logra obtener un reconocimiento exitoso o fallido. Si la imagen no es apropiada, el algoritmo por robusto que sea no alcanza a dar los resultados esperados [4].

Al manipular imágenes digitalizadas se trabaja a nivel matricial con la ayuda de los pos principios del álgebra lineal, es decir con operaciones matriciales. Para ello se requiere del uso de software matemático-científico para realizar operaciones a nivel matricial. Se utilizó el lenguaje

de programación Python y el interface de usuario "Spyder", que integra funciones y módulos avanzados para la computación matemática y científica.

La constitución de un sistema de visión artificial aplicado a la clasificación de objetos consta principalmente de tres bloques importantes: el bloque de pre-procesamiento, el bloque de segmentación y el bloque de conteo de objetos. Durante el pre-procesamiento se lee la imagen y se obtiene la matriz tridimensional con vectores de intensidad de pixeles en los canales rojo, verde y azul (RGB), adicionalmente se mejora la calidad de la información de los vectores eliminando el fondo y sobras, definiendo de esta forma al objeto dentro de una región de interés [2].

B. Preparación de los datos

Al digitalizar las imágenes en matrices numéricas estas se deben normalizar a un ancho estándar, que en este caso será de 500 pixeles de ancho. El redimensionamiento de imágenes es parte de los procesos de transformación de la imagen. Para redimensionar una imagen sin distorsionar su aspecto se utiliza el parámetro "ratio" designado con la letra (r)

El valor del ratio se puede calcular mediante la ecuación:

$$r = \frac{nuevo\ ancho}{ancho\ actual}. \tag{1}$$

Las dimensiones de re-dimensionamiento para la imagen serían: $[(nuevo\ ancho) \times (alto\ actual \times r)]$

De esta forma las imágenes digitalizadas tendrán un ancho estándar y se facilitará su manipulación matricial.

Se debe tomar en cuenta al momento de adquirir las imágenes que estas tengan un contraste equilibrado con la ayuda de una buena iluminación para no perder información relevante. Se realiza una ecualización del histograma para mejorar la calidad de las imágenes y mejorar la información en la extracción.

C. Extracción de características

La extracción de características de una imagen se puede realizar a través de algunos cuantificadores como: textura, color, forma, tamaño, cantidad, histogramas, etc.

El color de un pixel es una imagen se expresa con tres coordenadas en un espacio de color. Los más amplia mente utilizados en ordenadores e imágenes digitales so los espacios basados en los colores primarios rojo, verde y azul (RGB). Cuando los objetos inspeccionados tienen diferentes colores, a menudo una simple relación entre ellos puede discriminarlos, lo que ahorra tiempo de procesamiento [9].

Para el presente trabajo se implementa una técnica para la extracción de características en imágenes basado en el histograma de color "3D RGB".

Un histograma de color representa la distribución de intensidades en los pixeles que contiene una imagen ya sea de un color base o en escala de grises [10].

La gráfica del histograma de una imagen monocromática representa: en el eje horizontal los niveles de gris que pueden aparecer y en el eje vertical la altura en cada nivel de gris del histograma como el número de pixeles de la imagen que representan ese nivel de gris respectivamente. A este histograma se lo puede catalogar como un *histograma de color en una dimensión* [10].

La Fig.3 muestra el histograma en una dimensión de una imagen monocromática y los tres histogramas unidimensionales pertenecientes a una imagen de color sobrepuestos en un mismo gráfico.

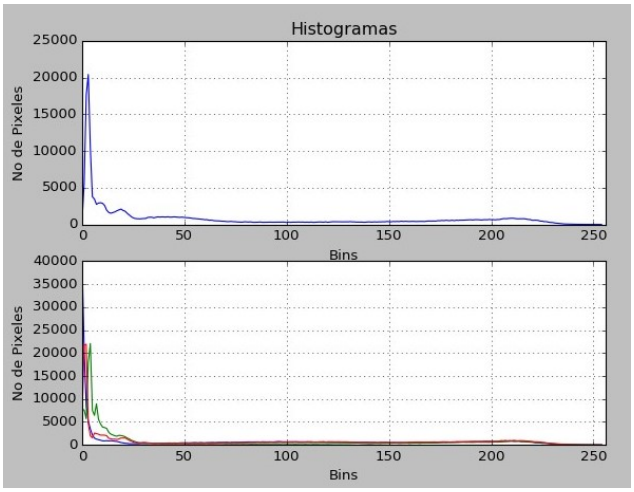

FIG. 3. En la parte superior se representa a una imagen monocromática, en la parte inferior tres imágenes monocromáticas sobrepuestas de cada canal pertenecientes a una imagen de color.

El histograma de la Fig.3 representa una imagen con predominancia de colores obscuros puesto que los picos más altos se encuentran a la izquierda en los niveles cercanos al cero.

Por otra parte, en un "histograma 3D RGB" el primer eje representa la escala de intensidades del color rojo (R), el segundo eje la escala de intensidades del color verde (G), el tercer eje representa la escala de intensidades del color azul (B), finalmente un cuarto eje representa el numero de pixeles de las intersecciones de las tres matrices [8].

Por su naturaleza un "Histograma 3D RGB" no se puede representar gráficamente puesto que se trata de cuatro dimensiones pero se puede procesar mediante matrices multidimensionales.

La característica del "Histograma 3D RGB" en cada imagen o clase de imagen es única y se refiere al color característico de cada imagen.

Extraer la característica de un objeto único en una imagen requiere una aplicación directa de la extracción del "histograma 3D RGB", pero extraer esta característica de un objeto en una imagen que contiene varios objetos se debe realizar mediante el enmascaramiento de la imagen.

El enmascaramiento aisla en la imagen el objeto o región de interés (ROI), es decir, una imagen aislada con un fondo negro. La Fig.4 muestra la imagen de una manzana, ya que el propósito de estudio es caracterizar el color de la manzana, la máscara resultante es la expuesta [10].

FIG. 4. A la izquierda se encuentra la máscara de la imagen, a la derecha la imagen enmascarada.

La información proporcionada por la extracción del "Histograma 3D RGB" de una imagen enmascarada proporcionará más información pertinente a los propósitos de estudio.

En la Fig.5 se puede apreciar la diferencia de información extraída de una imagen sin enmascarar y una enmascarada.

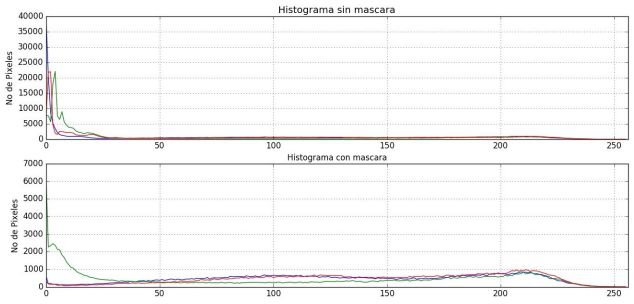

FIG. 5. En la parte superior está el histograma de una imagen sin enmascarar, en la parte inferior se encuentra el histograma de la misma imagen pero enmascarada

La creación de mascaras se puede efectuar automáticamente mediante algoritmos de umbralización de imágenes.

La umbralización o "Thresholding" es la binarización de imágenes, es decir determinar el valor (T) del color o brillo de los pixeles en un objeto de una imagen, luego transformar la imagen mediante la siguiente ecuación:

$$q = \begin{cases} 0 & \text{para} \quad p \leq T \\ 255 & \text{para} \quad p > T \end{cases} \tag{2}$$

Esta operación se puede interpretar como un sistema en donde la imagen de entrada con pixeles $p(x, y)$ ingresa al sistema y este entrega una imagen de salida o binarizada con pixeles $q(x, y)$.

La función de umbralización de la Ec.2 transforma los pixeles con un valor menor igual a T a cero, es decir de color negro y todos los valores mayores a T se transforman a un valor de 255, es decir color blanco, esto generará una máscara binaria determinada.

D. Máquinas de aprendizaje supervisado

Las máquinas de aprendizaje han evolucionado como un sub-campo de la Inteligencia Artificial y se clasifican de la siguiente manera:

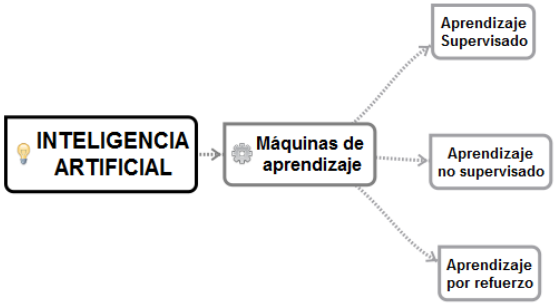

FIG. 6. Tipos de máquinas de aprendizaje.

Debido a la naturaleza de los datos del presente trabajo se utiliza una máquina de aprendizaje supervisado.

El objetivo principal en el aprendizaje supervisado es que un modelo aprenda a partir de un grupo de datos de entrenamiento etiquetados y luego nos permitirá hacer predicciones sobre datos ocultos o futuros. Aquí, el término supervisado se refiere a un conjunto de muestras donde ya se conocen las señales de salida deseadas (las etiquetas) [5].

La máquina de aprendizaje supervisado "aprende" de un grupo de datos previamente etiquetados para el entrenamiento. En otras palabras se ingresa un dato característico de una imagen a la máquina y al mismo tiempo se le da la respuesta, es decir el nombre de la imagen o "etiqueta", de esta forma la máquina aprende por iteración de varias imágenes [6].

Esta tarea aplicada a una máquina de aprendizaje se la conoce como "clasificación" puesto que se entrenan varias clases de imágenes para aplicar futuros reconocimientos o clasificaciones automáticas.

El procedimiento de aprendizaje supervisado aplicado en la clasificación de imágenes se resume a en la Fig.7

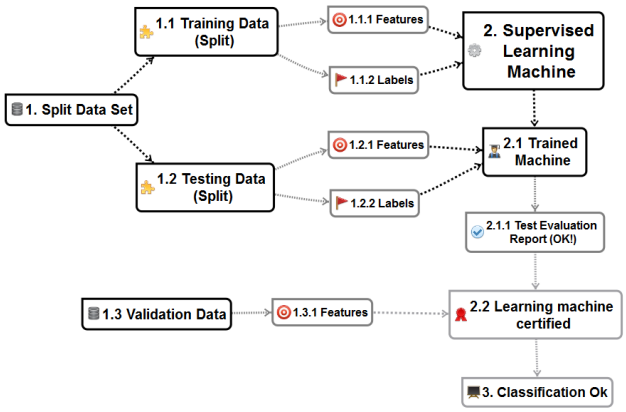

FIG. 7. Proceso del aprendizaje supervisado con imágenes.

El proceso de entrenamiento parte del conjunto de datos que es un vector de características del "Histograma RGB" de todas las imágenes etiquetadas, luego el conjunto de datos se divide en dos subconjuntos:"Training data" y "Testing data" respectivamente. El "Training data" entrena a la máquina de aprendizaje por medio del vector de características y su respectivo vector de etiquetas "Labels". Una vez entrenada la máquina se efectúan pruebas con el "Testing data" para obtener un reporte de la efectividad de las pruebas en la clasificación. Finalmente se dispone un conjunto de datos adicionales nombrado como "Validation data" que no es parte del conjunto de datos iniciales "Data set" y que sirve para validar el funcionamiento final del clasificador de imágenes [6].

El modelo de clasificador utilizado para esta tarea es "Random Forest" que está basado en árboles de decisión y tiene un buen rendimiento en tareas de clasificación.

El funcionamiento de un árbol de decisión se puede explicar mediante la Fig.8

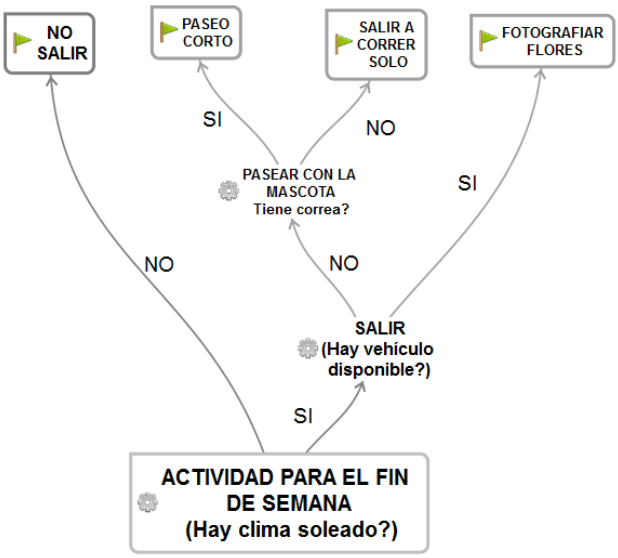

FIG. 8. Funcionamiento de un árbol de decisión.

En la Fig.8 para entrenar la máquina se ingresa un dato (la raíz) y al mismo tiempo se expone su etiqueta (hoja superior final) el algoritmo "Random Forest" genera internamente nodos de decisión y ramificaciones hasta llegar a la hoja final. Esto es un proceso iterativo que crea muchas opciones para la clasificación de clases de datos (diferentes hojas finales).

Un solo árbol de decisión podría generar funciones bastante complejas y por esto ser propenso a funcionar bajo reglas excesivas de aprendizaje generadas que funcionan solo con el conjunto de datos de entrenamiento. Para evitar esto se puede limitar el número de reglas que aprende, como por ejemplo limitar el número de capas de un arbol a tres. Este árbol aprenderá las mejores reglas para dividir el conjunto de datos a nivel global, y no aprenderá reglas muy específicas que separarían al conjunto de datos en grupos

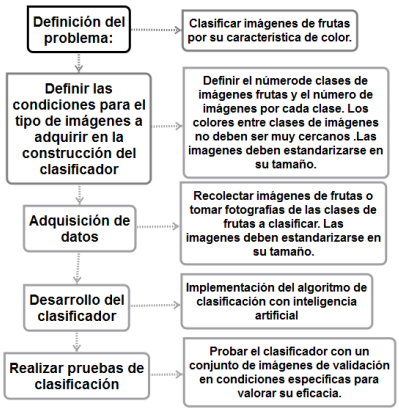

FIG. 9. Metodología propuesta en el desarrollo del trabajo.

altamente precisos. Esta compensación da como resultado árboles que poseen una buena generalización, pero un rendimiento más pobre. Se compensa esto generando muchos arboles de decisión y que luego cada uno prediga el valor de la clase. Luego se toma un voto mayoritario y usa esa respuesta como nuestra predicción general. Los bosques aleatorios trabajan en este principio [7].

En definitiva la máquina de aprendizaje supervisado "Random Forest" combina varios arboles de decisión para formar uno más grande y robusto.

El algoritmo Random Forest"se encuentra implementado en el módulo para python "scikit-learn" y con el nombre: "RandomForestClassifier".

III. MATERIALES Y METODOLOGÍA DE TRABAJO

A. *Materiales*

Los materiales utilizados son:

1) Cámara fotográfica digital con flash y auto enfoque:
2) Memoria SD de 8G clase 10
3) Computador Laptop PC con conexión a internet

B. *Metodología*

La metodología propuesta para el desarrollo del clasificador de imágenes de frutas se puede apreciar en la Figura Fig.9

Para solucionar la problemática de clasificación de imágenes de frutas se toma en cuenta los siguientes aspectos:

1) Se requiere automatizar el proceso de clasificación de imágenes de fruta para una variedad de aplicaciones ya sea en tiempo síncrono o asincrónico.
2) La clasificación va depender exclusivamente de las características del color de la fruta, ya que se debe definir de antemano los objetos a analizar.

El conjunto de imágenes adquiridas consiste en una recolección de imágenes de internet y de elaboración propia. El conjunto completo resulto en diferente número de

```
1  # Importar las librerías necesarias
2  import argparse
3  import cv2
4  import glob
5  import numpy as np # Librería para trabajar con matrices y vectores
6  # 1. Crear un objeto para el paso de argumentos
7  pasar = argparse.ArgumentParser()
8  # 2. Definir el nombre del argumento temporal de acceso
9  pasar.add_argument('-im')
10 # 3. Leer los argumentos de la línea de comandos y guardarlo
11 # en "argumentos". La función vars(), convierte el argumento
12 # en una variable de cadena de caracteres.
13 argumentos = vars(pasar.parse_args())
14 # Lee y graba las direcciones de las imagenes en una lista y
15 # con "sorted()"  se ordenan los archivos
16 dir_imagenes = sorted(glob.glob(argumentos['im'] + '/*.png'))
```

FIG. 10. Algoritmo que se encarga de importar las direcciones de las imágenes en formato matricial en código Python.

fotografías por cada una de las cuatro clases de frutas que en total suman ochenta y tres imágenes.

Debido a la variedad de fuentes de adquisición de las imágenes, estas se normalizaron a un formato y tamaño adecuados.

Se debe identificar en cada imagen las regiones de interés (ROI) y elaborar las máscaras respectivas.

Se divide el conjunto total de datos en dos subconjuntos y se adquiere uno adicional:

1) Conjunto de datos de entrenamiento
2) Conjunto de datos de prueba
3) Conjunto de datos de validación

Se implementa el algoritmo de clasificación y se entrena a la máquina de aprendizaje "Random Forest" con el conjunto de datos de entrenamiento, luego se evalúa el clasificador con el grupo de datos de prueba. Una vez realizada la prueba se debe analizar el reporte de clasificación, si este tiene un porcentaje alto de recuperación en la clasificación de imágenes superior al noventa por ciento se debe probar el conjunto de datos de validación.

IV. DESARROLLO DEL CLASIFICADOR

1. Diseño

Para la construcción de clasificador se debe tomar en cuenta las siguientes implementaciones que nos permitirán operar con las imágenes:

Para cargar las imágenes se utilizó el siguiente algoritmo descrito en la Fig.10:

En la Fig.10, se importan las librerías necesarias de la línea 2 a la línea 5, las líneas 7, 9, 13 y 16, se encargan de cargar las direcciones de las imágenes contenidas en un directorio determinado, esto se logra al ingresar un comando inicial en el terminal de Ipython que ejecuta el algoritmo en un archivo (*.py) y con los directorios indicados de la siguiente forma por ejemplo: "%run archivo.py -im imagenes/frutas/"

En la Fig.11, en las lineas que van de la 20 a la 25 con un bucle "for" se leen las direcciones de las imágenes y se las guarda en el vector "dir-im-it", en la linea 25 se lee y convierte en cada iteración una imagen en un arreglo matricial.

```
20 for i in np.arange(0,len(dir_imagenes)):
21     # Cargar una dirección en cada iteración en "dir_im_it"
22     dir_im_it=dir_imagenes[i]
23
24     # Se lee cada dirección y se guarda en la variable "imagen"
25     imagen = cv2.imread(dir_im_it)
```

FIG. 11. Algoritmo que permite iterar sobre las direcciones de las imágenes de un directorio determinado y guardarlas en un vector de direcciones "$dir - im - it$".

```
9  # Iteramos con las direcciones de las imágenes
10 for dir_imag in dir_imag:
11     # Leer y guardar imagenes en cada iteración en "image"
12     image = cv2.imread(dir_imag)
13     # Calcular el "ratio" definido como:
14     # (nuevo ancho) / (ancho actual de cada imagen)
15      r = 500 / image.shape[1]
16     # Establecer las dimensiones de la nueva imagen de
17     # dimensiones: [(500) x (alto_acual*ratio)]
18     # Así se mantienen el aspecto original de la imagen
19     dim =(500, int(image.shape[0]*r))
20     # Aplicar el redimensionamiento con la función de OpenCV,
21     # resize(). Donde los campos necesarios son:
22     # resize(imagen, nuevas dimensiones, tipo de interpolación)
23     redim=cv2.resize(image,dim,interpolation=cv2.INTER_AREA)
24     # Guardar cada imagen redimensionada de cada iteración
25     # con el formato: "mismo nombre.jpg" se puede cambiar la ext
26     nombre=dir_imag.split(".")[-2]+'.jpg'
27     cv2.imwrite(nombre, redim)
```

FIG. 12. Algoritmo de Re-dimensionamiento de imágenes.

Se debe estandarizar las imágenes a un mismo tamaño, para esto se implementó el siguiente algoritmo basado en las operaciones de transformación básicas de la imagen como se ve en la Fig.12:

En la Fig.12, Se itera sobre el vector de las direcciones delas imágenes cargados previamente en los algoritmos de las figuras Fig.10 y Fig.11, donde en la línea 15 se calcula el parámetro ratio que permite re-dimensionar cada imagen a un ancho estándar de 500 píxeles por su proporcional de altura para mantener el aspecto de cada imagen y luego almacenarla y re-nombrarla en la misma carpeta destino con las líneas: 19, 23, 26 y 27.

Para generar las mascaras de las imágenes y poder resaltar las regiones de interés de cada imagen utilizaremos la implementación del algoritmo de Otsu con auto-detección del valor "tresholding" (T). Dicho algoritmo implementado en el lenguaje python se expone en la Fig.13:

En la Fig.13, en la línea 16 se itera sobre el vector de dirección y nombres de las imágenes guardadas previamente "dir-imag", y en la linea 17 se van cargando en cada iteración las imágenes, en la línea 19 se transforma cada imagen de color en monocromática. En la línea 20 se ecualiza la imagen mediante la técnica del histograma.

```
16 for dir_imag in dir_imag:
17     image = cv2.imread(dir_imag)
18     #Convertir la imagen a escala de grises
19     image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
20     image = cv2.equalizeHist(image)
21     # Se aplica un desenfoque borroso Gaussiano con un kernel de 9x9.
22     # Luego se define la fuente para escribir en la imagen
23     borrosogauss = cv2.GaussianBlur(image, (9,9),0)
24     # Se implementa el método de Otsu
25     T = mahotas.thresholding.otsu(borrosogauss) # se aplica a la imagen borrosa
26     # Se implementa el threshold con el valor detectado de T con el método de Otsu
27     (T,thresh) = cv2.threshold(borrosogauss, T,255,cv2.THRESH_BINARY)
```

FIG. 13. Generación de máscaras con el algoritmo de Otsu y detección automática del valor de umbralización T.

```
1   # Importando librerías necesarias
2
3   import cv2
4
5   class histo_RGB:
6           def __init__(self, bins):
7       # Inicializamos el número de bins a utilizar (resolución de cada canal de color R,G,B)
8       # Se debe inicializar por ejemplo como "hist_RGB([8,8,8])
9                   self. bins = bins
10
11          def describe (self, imagen, mascara = None):
12      # Calcula el histograma en 3D de la imagen RGB
13      histo = cv2.calcHist([imagen],[0, 1, 2], mascara,self.bins,[0, 256, 0, 256, 0, 256])
14
15      # Se normaliza el histograma para que las imágenes que tengan
16      # el mismo contenido, pero escalas diferentes tengan el mismo
17      # histograma
18      cv2.normalize(histo, histo)
19      # Retorna el histograma 3D normalizado como una matriz plana
        return histo.flatten()
```

FIG. 14. Algoritmo de extracción del vector de características del "Histograma 3D-RGB" de una imagen.

```
1   # Inicializar los vectores: para la lista de datos y las etiquetas
2   # de clase
3
4   datos = []; etiquetas = []
5   # Inicializando el objeto para el descriptor de las imágenes que se
6   # basa en el histograma en 3DRGB como caracteristica de cada imagen con
7   # un valor de 8 bins para cada matriz RGB
8   desc = histo_RGB([8, 8, 8])
9   # Iteramos sobre los directorios guardados de las imágenes y sus
10  # mascaras sobre cada par de imágenes (imagen y máscara) para extraer
11  # el vector de características de cada una y almacenarlas en data
12  # mediante la función append ()
13
14  for (di, dm) in zip(dir_imag, dir_masc):
15          # Cargamos la imagen y su máscara actuales
16          image = cv2.imread(di)
17          mask = cv2.imread(dm)
18          mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
19          # Extracción de las características de cada imagen iterada
20          caracteristica = desc.describe(image, mask)
21          # Actualizamos cada imagen y sus etiquetas en un solo vector "datos"
22          datos.append(caracteristica)
23          etiquetas.append(di.split("_")[-2])
```

FIG. 15. Algoritmo de creación del vector de características y de etiquetas.

```
1   # Del vector de etiquetas se extraen los nombres únicos y se codifican
2   # las etiquetas con números del 0 al 5
3   etiquetasN = etiquetas
4   nombresu_etiquetas = np. unique(etiquetasN)
5   le = LabelEncoder()
6   etiquetasN = le.fit_transform(etiquetasN)
7   # Construyendo el "training set" y "data set"
8   # Dividimos el vector datos en "data" y construimos el "training set" y el "Testing set"
9   (trainData, testData, trainTarget, testTarget) = train_test_split(datos, etiquetasN,
                                                    test_size = 0.3, random_state = 41)
10  # Entrenar el clasificador con "trainData" y sus respectivas etiquetas "trainTarget"
11  model = RandomForestClassifier(n_estimators = 35, random_state = 41)
                                                    model.fit(trainData, trainTarget)
12  # Se evalúa el clasificador mediante un reporte de prueba del "Test set"
13  print(classification_report(testTarget, model.predict(testData), target_names =
                                                    nombresu_etiquetas))
```

FIG. 16. Algoritmo que codifica numéricamente las etiquetas, inicializa y evalúa la máquina de aprendizaje. Finalmente imprime un reporte de clasificación.

En la linea 23 se aplica un filtro de Gauss para suavizar la imagen. En la línea 25, previamente cargada la librería "mahotas" se accede a la función "tresholding.otsu()" y se extrae de la imagen el valor de umbralización tresholding "T". Finalmente en la línea 27 se crea la máscara de cada imagen y se almacena en "thres", para posteriormente ser guardada en el mismo directorio con un nombre y formato determinado.

La característica principal que fundamenta la clasificación de imágenes de frutas es mediante la extracción del "Histograma 3D RGB"y la aplicación de la máquina de aprendizaje supervisado "Random Forest" en la clasificación. El algoritmo de extracción de características mediante el "histograma 3D-RGB" se puede apreciar en la Fig.14

En la Fig.14, Se definen dos funciones "histo-RGB()" y "describe()", donde en la en la línea 3 se importa la librería especializada para visión artificial "OpenCV" que proporciona algunas funciones de pre-procesamiento, procesamiento y de clasificación de imágenes. En la línea 6 y 9 se inicializa la variable de resolución de los canales de colores de las imágenes a operar. En la línea 11 se crea la función "describe()" en la que se deben ingresar dos parámetros de entrada: la imagen y su respectiva máscara en formato matricial. En la línea 13 se calcula el histograma 3D-RGB, mediante la función "calcHist()" donde se ingresan cinco parámetros: la imagen digitalizada, número de canales, la máscara, el número de bines en formato de lista y el rango cromático de cada canal de color, la extracción se almacena en la variable "histo". En la línea 18 se normalizan los valores y en la línea final se aplana esta matriz en un vector plano.

Vale recalcar que el presente proyecto se basa en un experimento controlado, puesto que en la adquisición de imágenes se puede modificar algunos parámetros externos que mejoren la calidad de la información como por ejemplo el brillo, el enfoque, la cantidad de imágenes, etc.

La creación de los vectores de características y sus respectivas etiquetas para cada imagen se realiza en el código de la Fig.15:

En la Fig.15, en la linea 4 se crean los vectores vacíos para las características y sus respectivas etiquetas de identificación, en la linea 8 se instancia el algoritmo para la extracción de características del histograma "3D-RGB", desde la línea 14 se realiza una doble iteración con los vectores que contiene las direcciones de las imágenes y sus máscaras respectivamente. En la línea 16 se lee cada imagen y en la línea 17 se lee su respectiva máscara, en la línea 18 convertimos cada máscara a escala de grises, en la línea 20 se realiza la extracción de la característica del histograma en "3D-RGB" de cada imagen enmascarándola y finalmente en las líneas 22 y 23 se van agregando en cada iteración las características de cada imagen con su respectiva etiqueta de identificación mediante la función "append()".

En la Fig.16, En la línea 3 se respalda el vector de etiquetas original, en la línea 4 mediante el comando "unique()" de la librería numpy "np.unique()" se extraen los nombres de las 5 etiquetas correspondientes a las cinco clases de frutas a trabajar. En la línea 5 se crea un objeto para codificar los nombres de las etiquetas con números. En la línea 6 se transforma el vector las etiquetas con nombres a etiquetas con un código de identificación numérico.

En la línea 9 y 10 se divide el conjunto de datos en: un conjunto de datos de entrenamiento y otro para prueba, cada uno con su respectivo vector de etiquetas numéricas, 30 % y 70 % del grupo de las imágenes en la división respectivamente.

En la línea 12 se implementa la máquina de aprendizaje "Random Forest" y se aplican los conjuntos de imágenes de

```
1  # Probando las imágenes del clasificador
2  for i in np.arange(0, len(dir_imag_valida)):
3      # Direcciones de las imágenes y sus máscaras de prueba
4      dir_ima = dir_imag_valida[i]
5      dir_masc = dir_masc_valida[i]
6      # Cargamos las imágenes y sus máscaras de prueba
7      image = cv2.imread(dir_ima)
8      mask = cv2.imread(dir_masc)
9      mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
10     # Extraer las caracteristicas de la imagen de la planta enmascarada a predecir
11     caracteris = desc.describe(image,mask)
12     #Aplicar el predictor del tipo de planta
13     flor = le.inverse_transform(model.predict(caracteris))[0]
14     print(dir_ima)   # imprime el directorio de la imagen a predecir
15     print("El nombre de esta planta es {}".format(flor.upper()))
16     cv2.putText(image,"Planta identificada:"+flor.upper(),(10,20),
17                                         font, 0.55,(255,255,255),1)
18     cv2.imshow("image", image)  # muestra la imagen
19     cv2.waitKey(0)          # pulse una tecla para continuar
20 # cierra la ventana final de imagen
21 cv2.destroyAllWindows()
```

FIG. 17. Algoritmo de validación del clasificador.

entrenamiento y prueba. En las líneas 15 y 16 se imprime el reporte de clasificación.

Finalmente en el código de la Fig.17, se efectúa la validación de las predicciones mediante el conjunto de imágenes de validación. Se itera sobre el vector de imágenes y sus respectivas mascaras para realizar la predicción y comparar el valor de la característica de cada imagen leída con alguna información establecida en el entrenamiento y si son iguales se presenta la predicción.

2. *Población*

Como ya se expuso, se experimentó con cuatro clases de imágenes de frutas de colores característicos diferentes. El número de imágenes por cada clase es variable y se definen el la tabla I.

No. Class	Fruit	Sample Size	%
1	Manzana roja	24	28.91
2	Manzana verde	18	21.69
3	Banana	25	30.12
4	Naranja	16	19.28
	TOTAL	83	100.0

TABLA I
NÚMERO DE IMÁGENES POR CLASE DE FRUTAS A CLASIFICAR.

3. *Entorno*

El entorno de adquisición de los datos es una recopilación de imágenes en la web y de fotografías de creación propia.

4. *Intervenciones*

Una vez adquirido el banco de imágenes se aplicó una estandarización de extensiones al formato (*.jpg) y se las re-dimensionó a un estándar de 500 píxeles de ancho. Se creó las máscaras del conjunto de imágenes "Data set" con el algoritmo de Otsu ya descrito.

5. *Software Utilizado*

El lenguaje de programación utilizado en la construcción del clasificador es "Python" implementado en la compilación "WinPython" que contiene el entorno interactivo de trabajo (IDLE) llamado "Spyder".

Por otra parte las librerías utilizadas son:

1) Numpy: Provee capacidades matemáticas y científicas especializadas en el trabajo matricial.
2) Matplotlib: Librería de Plotting o trazado.
3) Scikit-learn: Librería de aprendizaje de máquina.
4) mahotas:Librería intermedia especializada en Visión Artificial
5) OpenCV: Librería especializada para Visión Artificial.

V. RESULTADOS Y DISCUSIONES

Al implementar y ejecutar el algoritmo descrito en la Fig.7 y en la Fig.17, el algoritmo entrega el reporte de clasificación de la Fig.18

```
Terminal 1/A
im_frutas/imp\im_x_04.jpg
El nombre de esta planta es MANZANA.ROJA

In [5]: %run clasificador.py --imag im_frutas/dataset/ --imagp im_frutas/imp
              precision    recall  f1-score   support

      banana       1.00      1.00      1.00         7
manzana.roja       1.00      1.00      1.00        11
manzana.verde      1.00      1.00      1.00         2
     naranja       1.00      1.00      1.00         5

 avg / total       1.00      1.00      1.00        25
```

FIG. 18. Reporte de clasificación de "Random Forest".

El reporte de clasificación en su primera columna describe la etiqueta de la clase de fruta detectada, en este caso cuatro frutas. En la segunda columna se detalla la precisión del clasificador que describe la relación en cada clase de el número de etiquetas originales respecto al número de etiquetas clasificadas exitosamente. Se puede observar que la precisión en la clasificación es del ciento por ciento en cada clase. La tercera columna "recall" o recuperación, describe un porcentaje de clasificaciones exitosas de cada clase, y como se puede observar se recuperaron exitosamente todas las clasificaciones. La cuarta columna describe el parámetro "f1-score" que se puede interpretar como un promedio ponderado de la precisión y recuperación, es decir es como una escala de referencia que va de 0 a 1 o de 0 a cien por ciento la mejor clasificación. Finalmente la última columna "support" describe el total de datos de cada clase tomados del "dataset" para efectuar el test para el reporte. Aproximadamente el algoritmo toma aleatoriamente el 30 % del "dataset" se utiliza para el "testdata" [8].

Luego, el algoritmo toma las imágenes del set de validación y realiza una predicción o clasificación de imágenes no etiquetadas y entrega su correcta clasificación como se aprecian en las figuras: 19, 20, 21 y 22.

Las imágenes no etiquetadas se clasifican correctamente como lo indicó el reporte de clasificación. De esta forma, el clasificador esta entrenado y es capaz de identificar y

clasificar automáticamente cuatro clases de frutas por su color característico.

En algunas pruebas realizadas con frutas de similar color en los reportes se pudo observar una disminución de la precisión del clasificador debido a que se tratada de frutas con colores similares o cercanos.

Esto concuerda con algunos investigadores que afirman que para el proceso de clasificación de frutas existen diferentes técnicas basadas únicamente en atributos de color y forma, pero plantean que diferentes frutas pueden presentar valores similares de color y que esto provoca que no se tenga aún métodos robustos y efectivos para identificar y distinguir imágenes de frutas [11].

Por otra parte, con relación a otros trabajos de reconocimiento de frutas como los descritos en [11], en el presente trabajo se evaluó diversos tipos de clasificadores bajo diferentes tamaños de muestras y se obtuvo una eficiencia superior con el clasificador "Random Forest" y con un menor coste computacional de procesamiento.

EL presente trabajo se enfocó en la optimización de la clasificación de frutas basado en una sola característica que es la de la extracción del histograma "3D-RGB" pero en futuros trabajos se adicionará la optimización del trabajo con la característica de forma para robustecer el clasificador.

VI. Conclusiones y recomendaciones

El clasificador de imágenes de frutas se fundamenta principalmente en la extracción de características de la imagen y las máquinas de aprendizaje, esta combinación se puede generalizar a varias aplicaciones en el campo de la Visión artificial.

El proceso aplicado en la construcción del clasificador de imágenes de frutas se puede resumir en los siguientes pasos: Adquirir imágenes, pre-procesarlas, ajustar su brillo mediante normalización del histograma y re-dimensionarlas a un estándar común. Posteriormente se crea un "dataset" de las imágenes con sus respectivas máscaras, luego se divide el "dataset" en dos vectores de datos: de características (histograma 3D RGB) y etiquetas. Finalmente se debe aplicar los vectores de datos a la máquina de aprendizaje

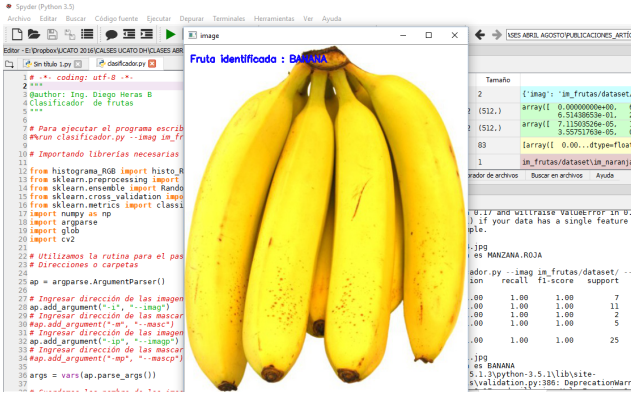

Fig. 19. Clasificación correcta de la fruta: "banana".

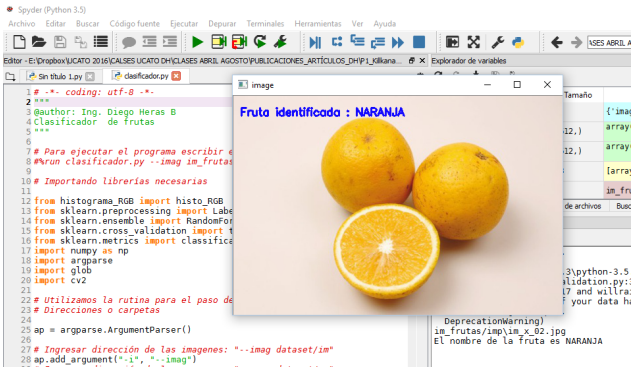

Fig. 20. Clasificación correcta de la fruta: "naranja".

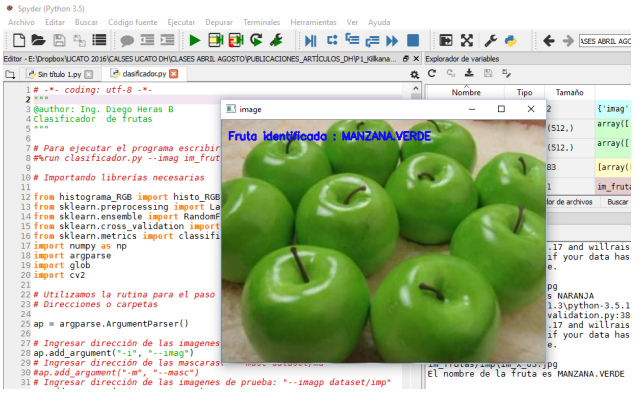

Fig. 21. Clasificación correcta de la fruta: "manzana verde".

supervisado. Para el entrenamiento asignar el 70 % de los datos y el 30 % restante para el test del reporte de clasificación (porcentajes que se ajustaron mejor al set de datos y se obtuvo los mejores resultados). Clasificar los datos del "set de validación" como prueba final.

El clasificador de imágenes de frutas, tendrá sus limitaciones al momento de entrenar dos clases de imágenes de frutas diferentes con colores muy cercanos entre si, para solucionar este inconveniente en futuros trabajos se complementará el clasificador con la extracción de otro vector adicional de características como por ejemplo de forma o textura para fortalecer el clasificador.

Un tema de aplicación que se puede potenciar más en futuros trabajos es sobre un procedimiento automático para la creación de máscaras, en el presente trabajo se utilizó el siguiente proceso: Digitalizar la imagen y aplicar un filtro Gaussiano para crear una versión borrosa de la imagen (suaviza los bordes), luego se determina el valor de "Tresholding" (T) mediante el método automático de "Otsu". Finalmente se aplica la "Umbralización" con el valor de T encontrado y se invierten los colores binarios de la máscara resultante.

La máquina de aprendizaje clasifica las imágenes según el color más cercano al que se le entrenó, por ejemplo: si introducimos en el conjunto de imágenes de validación una

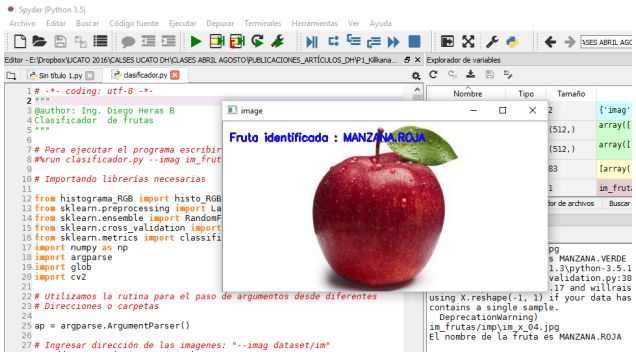

FIG. 22. Clasificación correcta de la fruta: “manzana roja”.

imagen de una mandarina, se clasificará seguramente como una naranja por la cercanía de sus características de color. Se debería entrenar previamente imágenes de mandarinas y comprobar si difieren el al reconocimiento con una naranja y en que grado.

En trabajos futuros los algoritmos implementados en el “Clasificador de imágenes de frutas” se pueden generalizar a la construcción de varios tipos de clasificadores en los que los objetos se puedan diferenciar por la característica del color y forma, como por ejemplo automatizar la clasificación de piezas de una empresa que sean de distintos colores y formas en sus materiales de construcción o de acabado, también implementar un clasificador para reconocer el estado de descomposición de frutas por su color, o un clasificador de especies de flores por su color y forma o textura, etc.

Como proyección se optimizarán los algoritmos de visión artificial implementados en este trajo en el área del “Deeep Learning” que mediante bibliotecas pertinentes se proporcionará mejoras considerables en las clasificaciones de las imágenes ya que los algoritmos se pueden ejecutar en potentes GPUs modernas tipo CUDA, como por ejemplo NVIDIA cuDNN.

REFERENCIAS

[1] C. Sanchez, N. Arizcuren & A. Casp,*Importancia de la visión artificial aplicada a la industria agroalimentaria*,Tecnología de Alimentos. Escuela Técnica Superior de Ingenieros Agrónomos. Universidad Pública de Navarra. Campus Arrosadía. 31006 Pamplona.

[2] L. I. Larcher, P. M. Juárez, A. I. Ruggeri, E.M Biasoni & G.A. Villalba,*Ponderación de calidad en frutas usando técnicas de visión artificial para la estimación de daños*, Mecánica Computacional Vol XXXII: 2473-2484.

[3] J. F. Vélez Serrano, A. B. Moreno Díaz, A. Sánchez Calle & J. L. Esteban Sánchez-Marín, J. L., *Visión por computador*, 2da. Ed. 2003.

[4] A. M. Romero, A. Marín-Cano & J.A. Jiménez-Builes,*Sistema de clasificación por visión artificial de mangos tipo Tommy*, UIS Ingenierías, enero-junio 2015; Facultad de Ingenierías Fisicomecánicas, UIS: 22-31.

[5] S. Raschka, *Python Machine Learning*, 1ra. Ed. UK: Packt Publishing Ltd., 2015.

[6] W. Richert y L. Coelho, *Building Machine Learning Systems with Python*, 2da. Ed. UK: Packt Publishing Ltd., 2015.

[7] R. Layton, *Learning Data Mining with Python*, 1ra. Ed. UK: Packt Publishing Ltd., 2015.

[8] P. Joshi y V. Godoy *OpenCV with Python By Example*, UK: Packt Publishing Ltd, 2015.

[9] S. Cubero, *Diseño e implementación de nuevas tecnologías basadas en visión artificial para la inspección no destructiva de la calidad de fruta en campo y mínimamente procesada*, Valencia, Mayo 2012: Tesis doctoral, 2012

[10] J. Ponce, *Computer Vision A MODERN APPROACH*, 2da. Ed. England: Pearson Education, 2015.

[11] C. M. Holguín, J. A. Cortés & J.A. Chaves.,*Sistema automático de reconocimiento de frutas basado en visión por computador*, Ingeniare. Revista chilena de ingeniería, vol.22 No 4, 2014, pp.504-516.

***Recibido**: 21 de junio de 2017*

***Aceptado**: 30 de agosto de 2017*

***Diego Heras:** Magister en Ingeniería Computacional y Matemática con 4 años de experiencia docente universitaria dictando las cátedras de Cálculo diferencial, Cálculo Integral, Ecuaciones Diferenciales, Métodos Numéricos, Estadística, Estática, Física Universitaria.*