ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SISTEMA DE CLASIFICACIÓN DE GRANOS DE CACAO FRESCOS BASADOS EN VISIÓN COMPUTACIONAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

ANGEL JEFFERSON OÑA OÑA

angel.ona@epn.edu.ec

DIRECTOR: Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO

felipe.grijalva@epn.edu.ec

CODIRECTOR: Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA

robin.alvarez@epn.edu.ec

Quito, enero 2020

AVAL

Certificamos que el presente trabajo fue desarrollado por Angel Jefferson Oña Oña, bajo nuestra supervisión.

Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO
DIRECTOR DEL TRABAJO DE TITULACIÓN

Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, ANGEL JEFFERSON OÑA OÑA, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

ANGEL JEFFERSON OÑA OÑA

DEDICATORIA

Este trabajo está dedicado a:

A mi madre Azucena y a mi padre René, por
ser mi inspiración, mi ejemplo de vida, por
estar en todo momento apoyándome, les agradezco
eternamente la paciencia que me han
tenido en todo este tiempo que ha durado mi carrera.

A mi sin igual familia por la confianza
en cada uno de mis proyectos académicos,
por llenarme de fortaleza
para culminar con lo emprendido.

A todos quienes demostraron
ser apoyo durante esta noble carrera.

Angel Jefferson

AGRADECIMIENTOS

A mis padres, por su entrega incalculable para alcanzar con éxito cada uno de mis objetivos, por ser el motor fundamental durante toda mi carrera, quienes con su ejemplo de perseverancia no permitieron rendirme durante este proceso.

A mi tutor PhD. Felipe Grijalva por su noble trabajo, como guía durante la ejecución de este sueño emprendido, por su paciencia y sabiduría entregada.

A mis demás familiares por estar presentes en cada triunfo, por confiar en mí y extenderme su mano cuando más lo he necesitado.

A mis maestros por todos los conocimientos impartidos, por guiar mi formación académica y demostrar siempre su apoyo incondicional.

A la Escuela Politécnica Nacional por ser cuna de conocimientos.

ÍNDICE DE CONTENIDO

RESUMEN

En la industria del cacao, especialmente en las pequeñas fincas productoras de cacao, los agricultores realizan la clasificación de los granos de cacao frescos con pulpa de una manera tradicional, es decir, a través de sus sentidos. Además, ellos no aplican ningún tipo de tecnología que acelere su producción.

En vista de ello, en el presente trabajo se propone un enfoque de clasificación de los granos de cacao frescos basados en visión computacional para los agricultores, el cual será muy útil en el proceso de remoción de la pulpa de los granos de cacao, permitiéndoles estimar eficientemente la calidad de estos granos.

Con este objetivo en mente, se describió una metodología para clasificar los granos de cacao frescos con pulpa utilizando el método k-means clustering y operaciones morfológicas para remover el fondo de las imágenes, luego se empleó un enfoque de Bolsa de Palabras Visuales (*Bag of Visual Words*, BoVW) como extractor de características y finalmente se clasificó estas características utilizando un clasificador de Máquinas de Vectores de Soporte (*Support Vector Machine*, SVM).

Como resultado, se alcanzó un Área Bajo la Curva ROC (*Area Under the ROC Curve*, AUC) de 99.68 % y una exactitud de 97.57 % manteniendo una tasa baja de falsos positivos de 2.46 %, lo que demuestra la factibilidad de usar visión computacional para este trabajo.

Nuestra base de datos de imágenes de granos de cacao frescos tiene 247 imágenes, donde 125 son de excelente calidad y 122 de mala calidad. Estas imágenes fueron etiquetadas por productores experimentados de cacao según su conocimiento sensorial.

PALABRAS CLAVE: k-means, BoVW, SVM, granos frescos de cacao con pulpa.

ABSTRACT

In the cocoa industry, especially in small farms, farmers perform the classification of fresh cocoa beans with pulp in a traditional way, i.e. through their senses. Moreover, they do not apply any kind of technology that makes their production faster.

Seeing that, in this work we propose a fresh cocoa beans classification approach based on computer vision for farmers, which will be very helpful in the process of removing cocoa beans pulp allowing them to estimate efficiently the quality of these beans.

With this aim, we describe a methodology to classify fresh cocoa beans with pulp by using the k-means clustering method and morphological operations to remove the background from the images, then we use a BoVW (Bag of Visual Words) approach as feature extractor and finally we classify these features using a SVM (Support Vector Machine) classifier.

As a result, we achieved an AUC (Area Under the Curve) of 99.68 % and an accuracy of 97.57 % by keeping a low false positive rate of 2.46 %, which demonstrates the viability of using computer vision for this task.

Our dataset of fresh cocoa beans pictures has 247 images, where 125 are excellent quality and 122 are poor quality. These images were labeled by experienced cocoa producers according to their sensory knowledge.

KEYWORDS: k-means, BoVW, SVM, Fresh cocoa beans with pulp.

1. INTRODUCCIÓN

El cacao, científicamente llamado Theobroma cacao, es una fruta tropical que necesita ciertas condiciones ambientales para cultivarlo. Además, solo hay ciertos lugares cuya posición geográfica presenta las condiciones ideales para cultivar esta fruta [1]. Ciertamente, Ecuador es un país que se ha posicionado como el principal productor mundial de cacao fino debido a su posición geográfica privilegiada, así como también por su abundancia de recursos biológicos [2].

Hoy en día, en las pequeñas fincas productoras de cacao, los agricultores utilizan técnicas experimentales para determinar la calidad del cacao. En su mayoría realizan este proceso a través de las propiedades organolépticas del cacao.

En general, estas propiedades describen las características físicas de la materia de acuerdo a cómo se las pueden percibir a través de los órganos de los sentidos, por ejemplo el sabor, la textura, el olor, el color o la temperatura de un objeto específico. Asimismo, este procedimiento se realiza sin utilizar ningún instrumento de medición [3], lo que significa que no hay estandarización en la medición de la calidad del producto.

Igualmente, la producción de cacao en las pequeñas y medianas empresas está sujeta a procesos artesanales mayoritariamente. Esto significa que hay una intervención predominante de la fuerza laboral de los agricultores desde etapas tan tempranas como la cosecha de las mazorcas o vainas de cacao hasta la fase de exportación.

Sin embargo, esta forma tradicional de clasificar el cacao no resulta ser la más adecuada ya que los pequeños y medianos agricultores utilizan más tiempo y cantidad de mano de obra en la ejecución de estos procesos en comparación con una empresa que implementa algún tipo de tecnología innovadora.

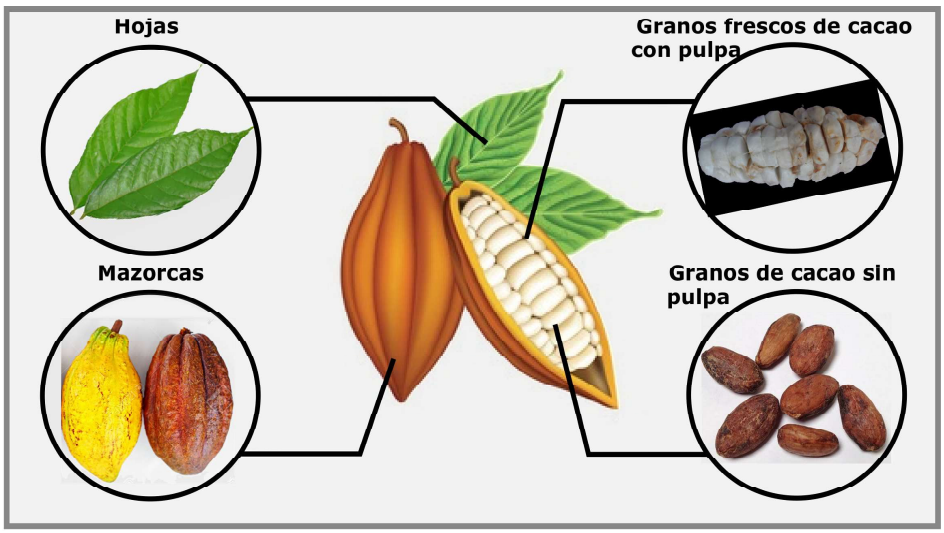


Figura 1.1: Diferencia entre mazorcas de cacao, granos frescos de cacao con pulpa (en baba) y granos de cacao sin pulpa.

Inclusive, los agricultores que emplean estos métodos sensoriales tradicionales, que tienen una naturaleza subjetiva, podrían introducir un producto dañado en el mercado del cacao. Por ejemplo, granos de cacao en buenas y malas condiciones mezclados. Consecuentemente, esto influirá en la calidad del cacao y gestará una reducción en las ganancias de los agricultores.

Para identificar claramente las vainas, los granos de cacao con pulpa y los granos de cacao sin pulpa, se muestran estos términos en la Figura 1.1.

De ahí que, por la gran importancia que tiene el cacao fino y debido a la necesidad de los agricultores de mejorar sus tiempos de producción disminuyendo el porcentaje de fruto dañado que será despulpado, en el presente trabajo se propone estimar la calidad de los granos de cacao frescos con pulpa [4] (ver Figura 1.1) de una forma confiable, segura y práctica, a través de métodos de visión computacional y un algoritmo de clasificación estadística aplicado a las imágenes digitales de los mismos.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es: Implementar un Sistema de Clasificación de Granos de Cacao Frescos Basados en Visión Computacional.

Los objetivos específicos del Proyecto Técnico son:

- Describir la información relevante acerca de la botánica, clases, producción e importancia del cacao.
- Estudiar el procesamiento de imágenes, operaciones morfológicas, algoritmo k-means destinado a la remoción del fondo y el modelo de extracción de características Bag of Visual Words (BoVW).
- Estudiar el principio de funcionamiento del aprendizaje supervisado.
- Desarrollar un script en Matlab que permita clasificar las imágenes de los granos de cacao frescos en excelente y mala calidad a partir de características extraídas con el modelo Bag of Visual Words (BoVW).
- Determinar el error de clasificación de los resultados mediante *k-fold cross-validation*.

1.2. ALCANCE

En este trabajo se clasificará una base de datos de fotografías de mazorcas de cacao de las cuales han sido removidas tanto su corteza externa como su corteza interna (ver Figura 1.2). Cabe mencionar que esta base de datos de los granos de cacao fue obtenida en el proceso de despulpado de esta fruta que se desarrolló en el proyecto de titulación de la UTEQ en la Facultad de la Carrera de Ingeniería Mecánica, disponible en la referencia [5].

Estas imágenes fueron clasificadas en excelente calidad, media calidad y mala calidad de acuerdo al conocimiento organoléptico de los agricultores de cacao. No obstante, en este proyecto de titulación se utilizarán únicamente las imágenes de excelente y mala calidad.

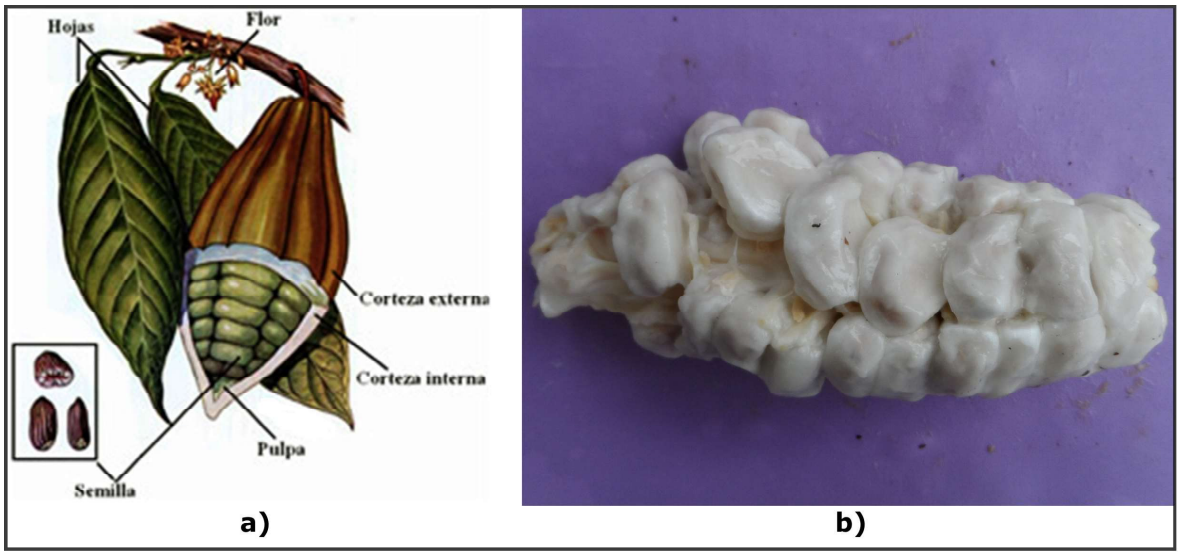


Figura 1.2: a) Partes de la planta de cacao; b) Granos de cacao frescos.

A continuación, en la Figura 1.3 se presenta el diagrama de bloques que representa en forma resumida el trabajo a realizar.

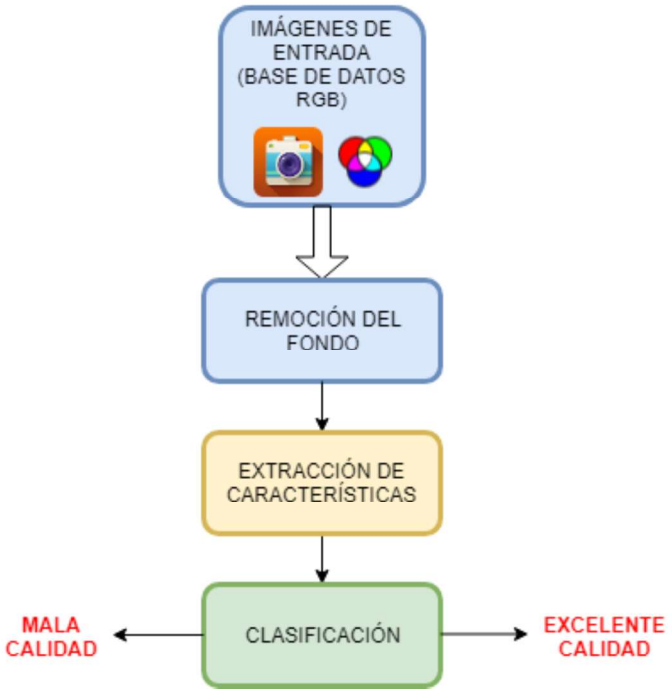


Figura 1.3: Diagrama de bloques simplificado del clasificador de imágenes a implementarse.

Primero, la imagen en Rojo, Verde y Azul (*Red Green Blue*, RGB) ingresará a un bloque de remoción de fondo, en donde se utilizarán operaciones morfológicas [6] y la técnica de *clustering* k-means [7] para que de esta forma se facilite el análisis del objeto centrándose

solo en los granos de cacao.

Luego, se tendrá un bloque que corresponde a la extracción de características de las imágenes del fruto de cacao. Para tal propósito se empleará el modelo BoVW (Bag of Visual Words) [8].

Seguidamente, se entrenará un clasificador que use como entradas las características obtenidas de BoVW y como salidas la etiqueta binaria de la imagen (i.e. EXCELENTE o MALA CALIDAD).

Finalmente, se evaluará la exactitud del sistema propuesto con k-fold cross-validation.

1.3. MARCO TEÓRICO

En este apartado se describirá la información relevante de tres principales temas incluyendo al final los trabajos anteriores. En cuanto a la primera Sección 1.3.1, se estudiarán ciertas Técnicas de Visión Computacional, en particular se hablará de las operaciones morfológicas, la segmentación con k-means y el modelo de Bolsa de Palabras Visuales (BoVW). Seguidamente, en la Sección 1.3.2, se describirá el Aprendizaje Supervisado con SVM, para lo cual se desarrollarán los conceptos de Aprendizaje de máquinas, Aprendizaje Supervisado y la Máquina de Vectores de Soporte (SVM). Por otro lado, en la Sección 1.3.3, se detallarán las métricas que se utilizó en este trabajo para evaluar el rendimiento del clasificador. Por último, en la Sección 1.3.4, se mencionarán a los trabajos anteriores que están relacionados con el presente proyecto y su diferenciación.

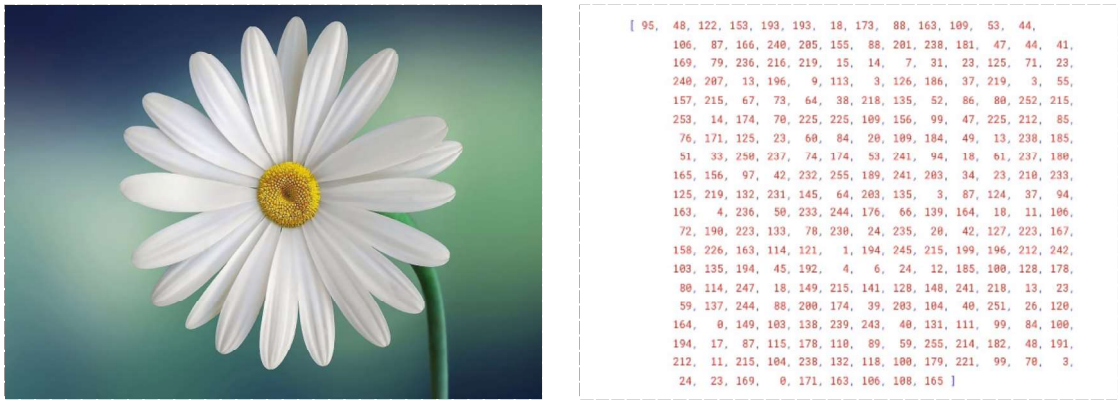
1.3.1. TÉCNICAS DE VISIÓN COMPUTACIONAL

La visión computacional denominada también como visión por computador, es la disciplina de la ciencia que se encarga de reproducir en una máquina lo más aproximadamente posible, la habilidad que tienen los ojos de analizar una imagen en el proceso tan complejo que implica la visión humana.

Igualmente, se puede decir que la visión computacional o comprensión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de ese mundo [9]. En el campo de la visión computacional, cuando se habla de las estructuras y propiedades del mundo tridimensional, se refiere tanto a sus propiedades geométricas como a sus propiedades materiales. Por ejemplo, las propiedades geométricas abarcan el tamaño y la forma de un objeto, mientras que las propiedades materiales corresponden al color, a la textura, la iluminación, entre otras.

Entender lo que se encuentra dentro de una fotografía no es tan sencillo como parece para una máquina. Entonces, analizando la Figura 1.4, el cerebro humano inmediatamente después de ver esa imagen puede saber que es una flor, pero esto sucede ya que nuestros cerebros han adquirido ese conocimiento durante varios años donde el contexto evolutivo

nos ayuda a entender al instante lo que es. Sin embargo, una computadora no tiene esa misma ventaja y para un algoritmo la imagen de la Figura 1.4.a, luce como en la Figura 1.4.b, es decir aquí no hay un contexto sino muchos datos que representan intensidades en el espectro de color.



a) b)

Figura 1.4: a) Interpretación de la imagen por un humano, b) interpretación de la imagen por una computadora.

En definitiva, mediante las técnicas de visión computacional [10] se adquiere información relevante de imágenes procedentes de algún dispositivo como una cámara o un celular, para posteriormente enviar esa información a un computador y que este lo pueda asimilar con facilidad.

1.3.1.1. Operaciones Morfológicas

La morfología matemáticamente se entiende como una herramienta de extracción de componentes de la imagen que son útiles en la representación y descripción de la forma de su región, como es el caso de los bordes, las zonas convexas, entre otros [6].

Las operaciones morfológicas son operaciones que se aplican originalmente sobre imágenes binarias, no obstante, se pueden también usar sobre imágenes en escala de grises o incluso sobre imágenes RGB.

El propósito de emplear operaciones morfológicas es cambiar ciertas regiones de la imagen, para que por ejemplo se resalten sus propiedades o en otras ocasiones se eliminen algunos objetos no deseados o propiedades que no se esperan en las imágenes resultantes.

Así pues, para aplicar las operaciones morfológicas se necesita de dos elementos:

- Imagen a la que se aplica la operación morfológica.
- Elemento estructurante.

Donde el elemento estructurante es un pequeño conjunto (un pequeño arreglo binario) o subimagen [6] que define la forma y el tamaño de la vecindad del pixel que será examinado,

para posteriormente alterar su valor. A estos elementos estructurantes se los considera también como máscaras, ventanas y algunos ya vienen predefinidos por MATLAB, por ejemplo: cuadrados, rectángulos, en forma de línea, en forma de círculo, en forma de diamante, etc.

En la Figura 1.5.a se muestran algunos ejemplos de elementos estructurantes y en la Figura 1.5.b se indican los elementos estructurantes convertidos a arreglos rectangulares [6]. Conviene señalar que cuando los elementos estructurantes van a trabajar con imágenes, se requiere que ellos sean unos arreglos rectangulares. Adicionalmente, hay que definir para cada elemento estructurante cuál es su centro o también llamado origen [6] teniendo en cuenta que usualmente pero no siempre ese origen está en la mitad de la forma del elemento estructurante que se esté usando. Este centro se puede apreciar en la Figura 1.5 como los puntos negros.

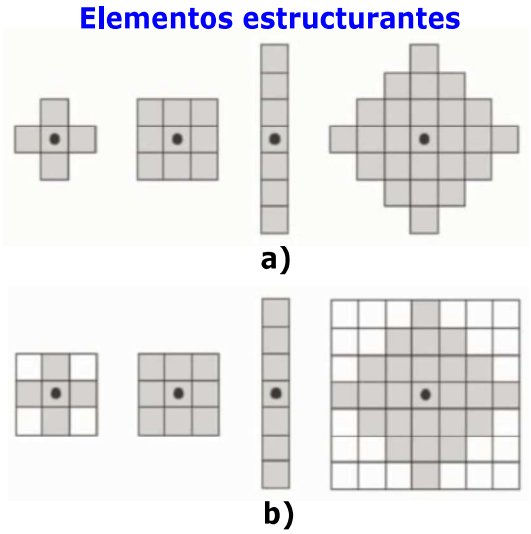


Figura 1.5: Ejemplos de elementos estructurantes. Adaptado de [6]

Particularmente, la operación de llenado de huecos se indica a continuación:

Llenado de huecos
Esta operación morfológica consiste en llenar los huecos de una imagen binaria de entrada. En este contexto, un hueco se define como el conjunto de píxeles del fondo que no se pueden alcanzar llenando el fondo desde el borde de la imagen [11].

Para ejecutar el llenado de huecos en una imagen, se tiene un algoritmo que está fundamentado en la operación morfológica dilatación y las operaciones de conjuntos complemento e intersección [12] (ver Ecuación 1.1).

Así pues, suponiendo que A representa a la imagen de entrada (ver Figura 1.6.a) cuyos elementos son 8 bordes conectados, cada borde encierra una región del fondo (i.e. un hueco). Dado un punto p denominado punto de partida que se refiere a un pixel dentro de la región antes mencionada como un hueco (i.e. la región rodeada por un borde de 8 conexiones), el propósito de llenar toda esa región con 1s se logra usando un procedimiento iterativo, matemáticamente expresado como la Ecuación 1.1.

En otras palabras, se empieza creando un arreglo $X_0 = p$ (ver Figura 1.6.d) y a través del procedimiento de la Ecuación 1.1 se llenan todos los píxeles pertenecientes al hueco con 1s.

$$X_k = (X_{k-1} \oplus B) \cap A^c \qquad k = 1, 2, 3, ... \tag{1.1}$$

donde B representa a un elemento estructurante simétrico (ver Figura 1.6.c). El algoritmo se detiene en la iteración k, i.e. cuando $X_k = X_{k-1}$. El conjunto X_k incluye a todos huecos llenados.

Finalmente, el resultado de la unión de X_k con A contiene al borde original (A) y a todos los píxeles dentro de él etiquetados con 1(ver Figura 1.6.i).

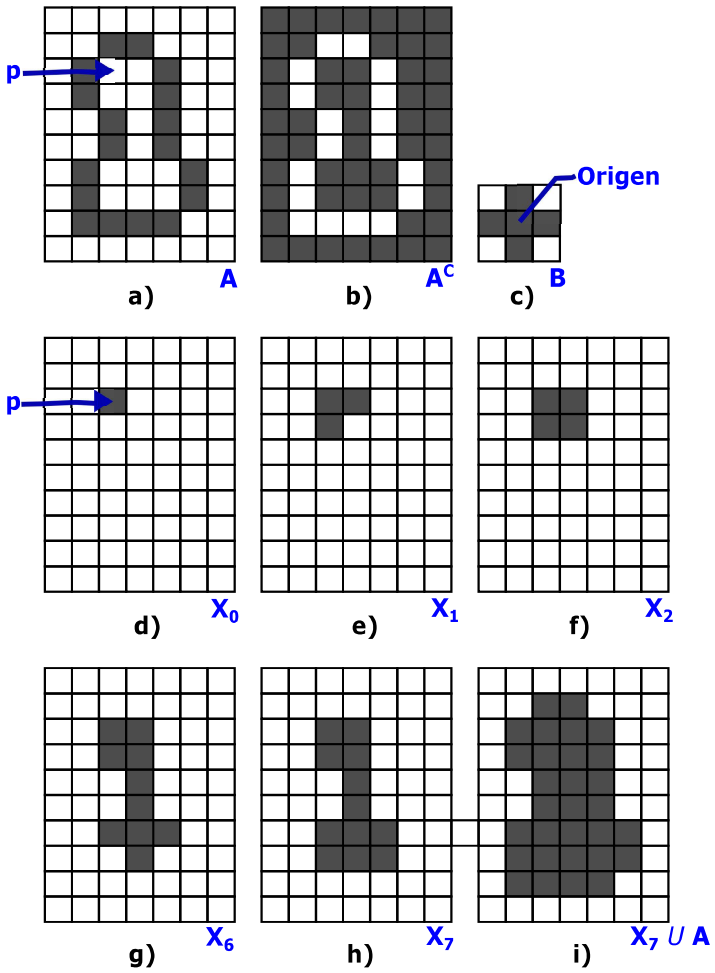


Figura 1.6: Aplicación de la operación morfológica llenado de huecos. Adaptado de [6]

Se puede observar en la Ecuación 1.1 que la parte de la operación morfológica dilatación, llenaría el área completa si no fuera controlada. Sin embargo, se puede apreciar que la intersección que se efectúa en cada paso con A^c (complemento de A, ver Figura 1.6.b) limita el resultado dentro de la región de interés [6]. El resto de gráficas en la Figura 1.6 ilustran el procedimiento mecánico de la Ecuación 1.1.

1.3.1.2. Segmentación con k-means

Ahora bien, k-means se refiere a una clasificación no supervisada (i.e. no posee una variable dependiente) la cual tiene como fin organizar los datos en grupos (*clusters*), de manera que los miembros de cada grupo sean similares entre sí en cuanto a que sus características estén más próximas y diferentes a las de otro grupo [7]. Dicho de otra manera, el algoritmo k-means intenta buscar en los datos, grupos con similitud de características, maximizando la variación *inter-cluster* (distancia entre los grupos representada en la Figura 1.7 con la línea en color amarillo) y minimizando la *intra-cluster* (distancia interna dentro de cada grupo representada en Figura 1.7 con la línea en color gris).

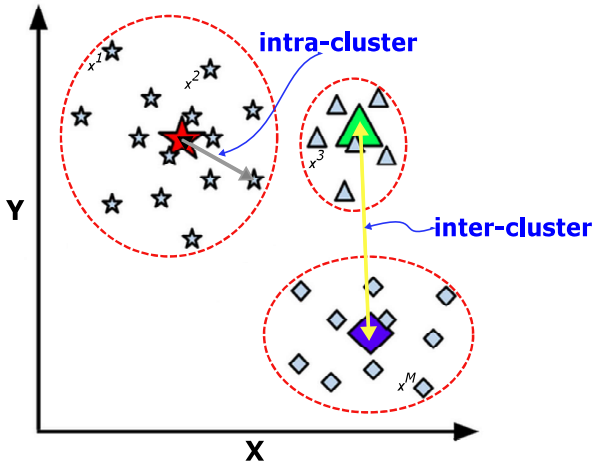


Figura 1.7: Distancias inter-cluster e intra-cluster en el algoritmo k-means. Adaptado de [13].

En síntesis, el algoritmo k-means se basa en los siguientes pasos:

Para la entrada k que indica el número de *clusters* y el conjunto de M datos sin etiquetar: $\{x^1, x^2, ..., x^M\}$

1. En primer lugar, se inicializa aleatoriamente los k centros. Estos centros son puntos o muestras que pueden estar dentro de los datos o pueden ser puntos nuevos.
2. Luego, se asigna cada muestra de los datos al centro más cercano.
3. Cuando todas las muestras hayan sido asignadas al centro correspondiente, se recalcula las posiciones de los centros de manera que se conviertan en el centroide de las muestras que tienen asignadas dichos centros. En esta parte, conviene señalar que hay diferentes métodos para calcular la distancia del centroide más cercano, uno de los más utilizados es la distancia Euclideana [7].
4. Seguidamente, se repiten los pasos anteriores hasta que los centros ya no se muevan (i.e. hasta que se empiece a obtener resultados repetidos en análisis consecutivos). Esto produce una separación de los datos en k grupos de manera que las muestras más parecidas entre sí pertenezcan al mismo grupo [7].

Por otra parte, es importante destacar que el algoritmo k-means tiene las siguientes particularidades:

- Es rápido.
- Es eficiente ya que solo se requiere almacenar los k centroides.
- El número de *clusters* debe ser especificado previamente.
- Los resultados finales son dependientes de la inicialización ya que esta es aleatoria.
- El algoritmo ya no es robusto cuando los datos están muy superpuestos o cuando se presentan valores atípicos (e.g. cuando una observación está muy distante del resto de los datos) [13].

Básicamente, k-means inicializa centroides al azar y permite encontrar los grupos, por ejemplo en la Figura 1.8 se está agrupando en Clúster A, Clúster B y Clúster C. En cada uno de esos grupos sus elementos son los más próximos al centroide, por eso k-means lo que analiza fundamentalmente es la proximidad.

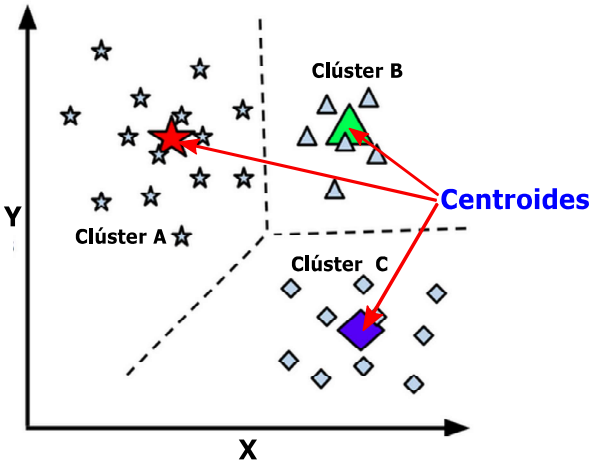


Figura 1.8: Ejemplo de la segmentación con k-means. Adaptado de [13].

1.3.1.3. Modelo de Bolsa de Palabras Visuales (BoVW)

La Bolsa de Palabras Visuales (*Bag of Visual Words*, BoVW) es una técnica que se emplea para construir descriptores de características locales, i.e. es un método de caracterización local [14] en donde se habla específicamente de puntos de interés.

El modelo Bolsa de Palabras Visuales (BoVW) inicialmente fue creado para la clasificación de texto y se conocía como *Bag of Words*, pero más tarde fue adaptado para trabajar también con imágenes. Este modelo es muy utilizado debido a su simplicidad, eficiencia y eficacia [8], i.e el procedimiento del modelo de Bolsa de Palabras Visuales es relativamente simple en comparación con técnicas muy complejas como *Deep Learning* y al ser un proceso simple implica que se ejecutará más rápido, por tanto, será un método eficiente. Además, BoVW tiene un alto grado de eficacia ya que posee buenos resultados de tasa de acierto cuando se lo utiliza con algún clasificador.

Por otra parte, el procedimiento del modelo Bolsa de Palabras Visuales (BoVW) se indica en la Figura 1.9 y está compuesto fundamentalmente de las siguientes etapas [8]:

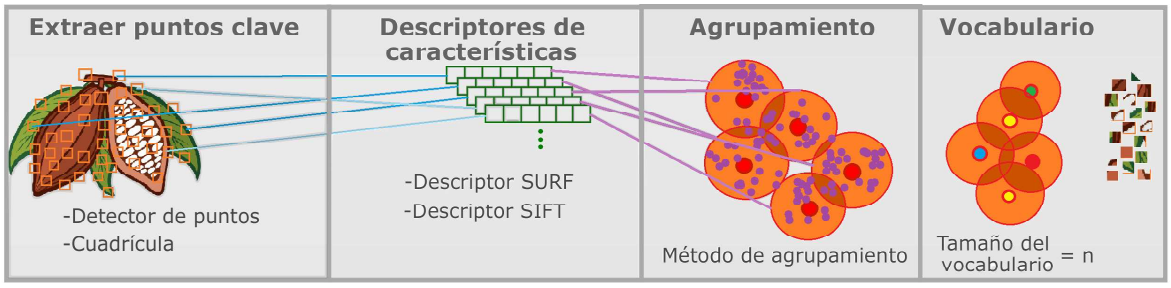


Figura 1.9: Representación del proceso de flujo de trabajo del algoritmo Bolsa de Palabras Visuales.

1. Extracción de puntos clave.
2. Generación de los descriptores de características.
3. Agrupamiento de los descriptores.
4. Construcción del vocabulario.
5. Representación del histograma de características.

Extracción de puntos clave

En primer lugar, se extraen los puntos clave o también denominados puntos de interés. Estos puntos son ubicaciones distintivas en la imagen, tales como esquinas o manchas [15]. Comúnmente, hay dos formas de realizar este paso, ya sea mediante:

- Una cuadrilla: Se refiere a una cuadrilla uniforme, cuyas intersecciones de las líneas que la conforman van a definir las ubicaciones de los puntos de interés [16].
- Un detector de puntos: Aquí se puede elegir el detector de puntos de Características Robustas Aceleradas (*Speeded Up Robust Features*, SURF [15]) o el detector de puntos de Transformada de Características Invariantes a la Escala (*Scale Invariant Feature Transform*, SIFT) [17]. Es importante destacar que la técnica SURF es más rápida que SIFT [18].

Generación de los descriptores de características

En esta parte, para cada punto de interés detectado en el paso anterior, el algoritmo analiza su vecindad y extrae de él un descriptor de características (es decir, un descriptor es la descripción de las características principales del punto clave). Igualmente, se tienen los siguientes tipos de descriptores:

- Descriptor SIFT: Es un descriptor de 128 dimensiones o características.
- Descriptor SURF: Utiliza un descriptor de 64 dimensiones que es invariante a la escala y rotación [15].

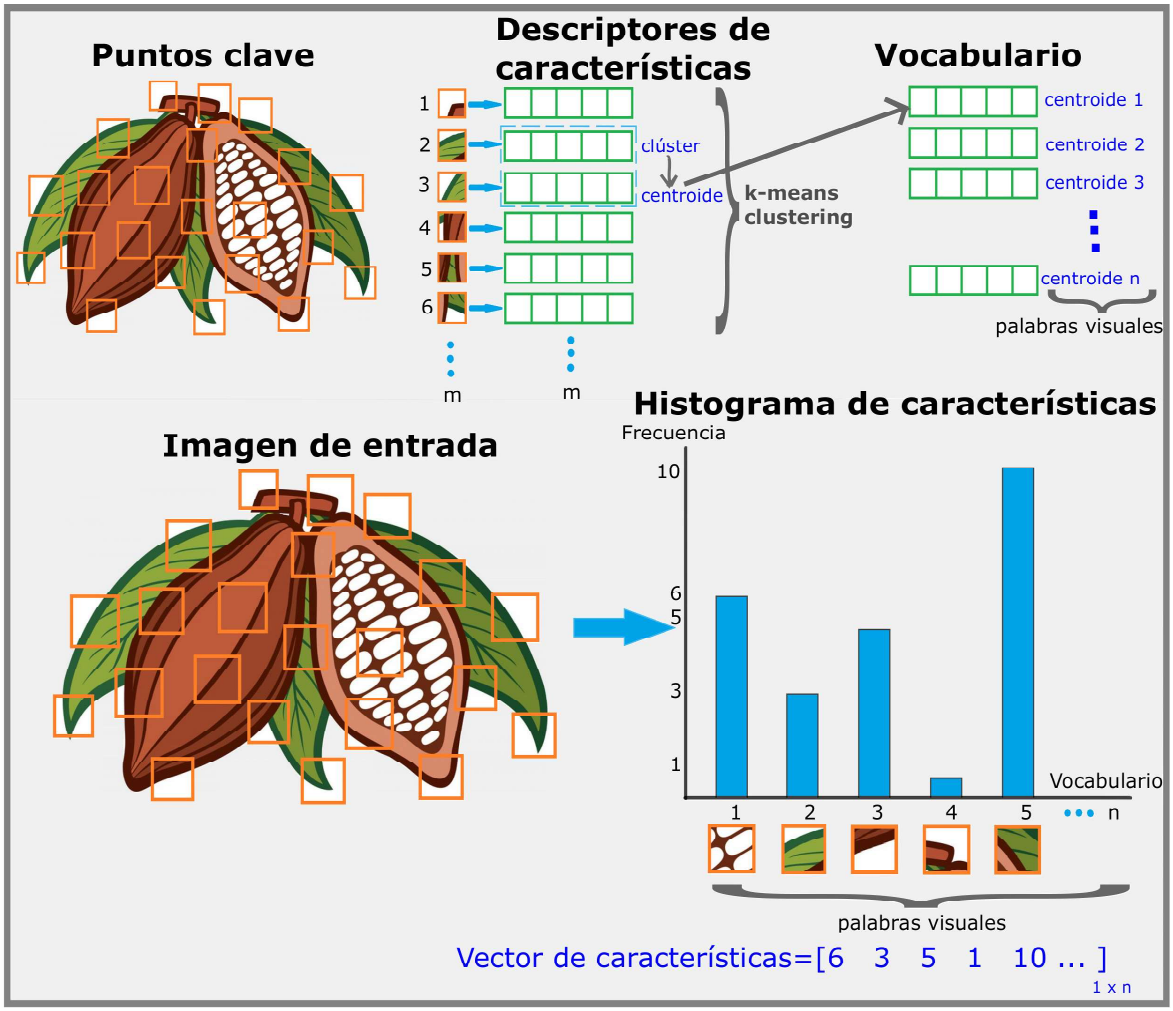


Figura 1.10: Obtención del histograma de características de una imagen de entrada y su vector de características correspondiente.

Se puede observar en la Figura 1.10 una cantidad *m* de puntos de interés, esta cantidad de puntos no siempre es la misma en cada imagen. Sin embargo, lo que sí es constante es el valor de la dimensión de los descriptores de características que hay por cada punto de interés seleccionado.

Agrupamiento de los descriptores

El proceso de agrupamiento (*clustering*) se utiliza para agrupar descriptores de características que sean similares. Por lo tanto, sobre los descriptores de características, se ejecuta el método de agrupamiento k-means para obtener *n* número de *clusters* (ver Figura 1.10).

Construcción del vocabulario

Una vez que se tiene estos grupos (*clusters*), se obtiene los centroides de cada uno de ellos. Como resultado, las palabras visuales (*visual words*) del vocabulario serán los centroides de esos *clusters* (ver Figura 1.10).

Asimismo, se puede observar en la Figura 1.10 que los k *clusters* corresponden al tamaño del vocabulario que es de valor *n* (i.e. *n* es el número de palabras visuales).

Representación del histograma de características
Cuando ya se dispone del vocabulario, para cada imagen se construye un histograma a partir de la frecuencia de las palabras visuales en la imagen, como se puede apreciar en la Figura 1.10. Los histogramas de características representan a los vectores de características que se proveen a un clasificador.

1.3.2. APRENDIZAJE SUPERVISADO CON SVM

1.3.2.1. Aprendizaje de Máquinas

Es habitual confundir los términos Inteligencia Artificial, Aprendizaje de Máquinas, Big Data o Deep Learning ya que muchas veces estos conceptos se solapan entre sí. Para ser específicos, el aprendizaje de máquinas o aprendizaje automático del inglés *machine learning* [19], es la rama dentro del campo de la Inteligencia Artificial que estudia como dotar a las máquinas de capacidad de aprendizaje entendido este como la generalización del conocimiento a partir de un conjunto de experiencias.

Dentro del aprendizaje de máquinas existen diferentes técnicas que sirven para resolver diversos tipos de aplicaciones, por ejemplo, los modelos de regresión, los modelos de clasificación, las técnicas de *clustering*, entre otros. Generalmente, los algoritmos de aprendizaje de máquinas tienen especificados los parámetros que controlan la tasa de aprendizaje o la capacidad del modelo subyacente [20]. Estos parámetros a menudo se consideran molestias que interfieren la ejecución eficiente de diferentes aplicaciones, por lo tanto, se puede considerar el gran interés y la atracción en desarrollar algoritmos de aprendizaje automático con un reducido número de parámetros.

Por otro lado, una forma más flexible de abordar el problema de los parámetros es considerar la optimización de ellos como un procedimiento para ser automatizado, y concretamente la mejor opción es la optimización bayesiana. Esta optimización es recomendada, por cuanto se ha demostrado que supera a otras optimizaciones globales de última generación algorítmica [20].

Los algoritmos de aprendizaje de máquinas tienen ciertas características que los distinguen de otros, en las cuales se puede mencionar que:

- Cada evaluación de la función requiere una cantidad determinada de tiempo, por ejemplo no es lo mismo entrenar una red neuronal pequeña con 10 unidades ocultas que una red más grande con 1000 unidades ocultas, situación que inclusive alterarán los costos de implementación.

- Los experimentos de aprendizaje automático a menudo se ejecutan en paralelo, en múltiples núcleos o máquinas. De esta manera se puede concluir que el aprendizaje automático o aprendizaje de máquinas se refiere a una disciplina encargada de la inteligencia artificial en donde se programa sistemas automáticos para identificar patrones de un conjunto de datos almacenados [20].

1.3.2.2. Aprendizaje Supervisado

Cuando se habla de aprendizaje supervisado se refiere a un subconjunto del aprendizaje de máquinas donde se intenta modelar las relaciones que hay entre unas variables de entrada y unas variables de salida i.e. los resultados, para realizar predicciones de esos resultados en nuevos conjuntos de datos [21].

El término supervisado viene del hecho de que al mostrarle los resultados esperados al algoritmo, estamos de igual forma siendo parte de la supervisión de su aprendizaje. Conviene destacar que las técnicas de aprendizaje supervisado realmente funcionan, porque basta con indicarle a uno de estos algoritmos, suficientes datos de entrada y de salida y si se encuentra alguna relación entre ellos, el algoritmo será capaz de aprenderla.

Puesto que los modelos de aprendizaje supervisado realizan predicciones tomando en cuenta datos del pasado (datos históricos), estos modelos necesitan ser reconstruidos periódicamente para que de esta forma sus predicciones no se vuelvan obsoletas [22]. Justamente, se debe considerar esto ya que en ocasiones el comportamiento de los datos puede variar.

Los algoritmos de aprendizaje supervisado tienen en común que se conoce de ellos sus características (datos de entrada) y lo que se pretende adquirir como resultado. No obstante, estos algoritmos se diferencian unos de otros ya que están orientados a resolver distintos problemas, de ahí que se puede dividir a los algoritmos de aprendizaje supervisado en los siguientes grupos:

- Clasificación.
- Regresión.

Clasificación

En esta área del aprendizaje supervisado, los algoritmos intentan hallar un modelo que permita describir la distribución de las clases, i.e. cómo se encuentran las clases repartidas en el espacio de características con el propósito de asignarles una clase. Una vez que se ha determinado el modelo, se lo utiliza para designar una clase a nuevos elementos.

Por ejemplo, un problema de clasificación puede ser el determinar si un carro es sedán o SUV a partir de sus características tales como, el tamaño, número de puertas, número de asientos, etc. Conviene destacar que las clases sedán y SUV son valores discretos o categóricos.

Regresión

En este caso, los algoritmos tratan de encontrar la curva que mejor se ajuste a los datos de entrenamiento y tras determinar esta curva, se la emplea en la predicción de valores nuevos, i.e. el algoritmo utiliza información histórica que luego nos posibilita predecir información futura o que desconocemos.

Un ejemplo para este caso sería un problema de predicción del precio de un carro en base a sus características (tamaño, número de puertas, número de asientos, etc). Lo importante que se puede notar aquí es que el precio es un valor continuo, un valor real.

1.3.2.3. Máquina de Vectores de Soporte (SVM)

La Máquina de Vectores de Soporte (*Support Vector Machine*, SVM) se define como un clasificador binario de algoritmo supervisado que se lo puede usar tanto en problemas de clasificación como de regresión. La idea básica de SVM es que si se tienen dos clases de datos que se encuentran etiquetados (generalmente se cambian las etiquetas a números, e.g. 1 y 2), el propósito es encontrar una recta o una superficie (hiperplano) que separe a las dos clases.

De esta manera, con este algoritmo se busca el hiperplano que separe las clases de la forma más correcta, es decir, teniendo el mayor margen o espacio de separación entre las observaciones [23]. Un clasificador como es el caso de SVM, toma a los vectores de características como entradas y determina en la salida a qué clase pertenecen.

El rendimiento de un clasificador SVM es dependiente de la función de *kernel* que se elija [24]. Entonces, el *kernel* se encarga esencialmente de tomar las muestras, sus características y transforma a estos datos de manera que los pueda dividir de la manera más acertada, así como se puede observar en la Figura 1.11.

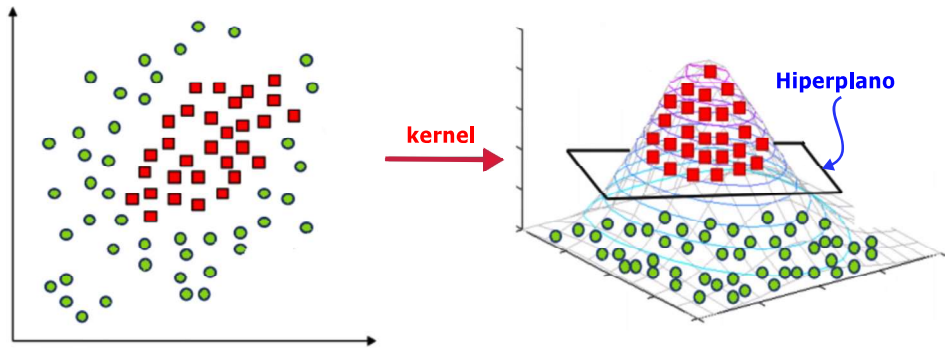


Figura 1.11: Ejemplo de funcionamiento del *kernel*. Adaptado de [24].

A continuación se presenta a la función de núcleo lineal (*linear kernel function*) y a la función de núcleo gaussiano (*gaussian kernel function*), mediante la Ecuación 1.2 y la Ecuación 1.3 respectivamente:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2 \tag{1.2}$$

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2) \tag{1.3}$$

donde $\mathbf{x}_1, \mathbf{x}_2$ son un par de vectores de características del conjunto de entrenamiento.

Cuando las clases $\mathbf{x}_1$, $\mathbf{x}_2$ no son separables, el objetivo principal del algoritmo no cambia, es decir, se trata de obtener el mayor margen posible. Sin embargo, el algoritmo introduce un tipo de penalización en cada clase, esta penalización está controlada por una constante C que actúa como un parámetro de regularización para evitar el sobreajuste.

Por el contrario, cuando se pueden separar las clases, hay varias soluciones, pero la ideal es aquella que maximice las distancias entre las muestras.

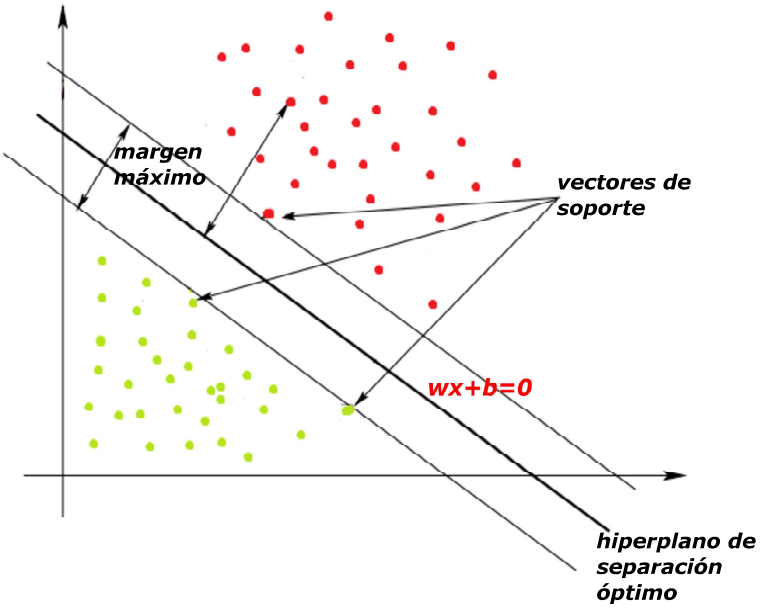


Figura 1.12: Hiperplano de margen máximo para un clasificador SVM con muestras de dos clases. Adaptado de [26].

El algoritmo intenta buscar los vectores de soporte que maximicen esa distancia [25]. Entonces, como se puede ver en la Figura 1.12, la mejor separación es la función lineal de la Ecuación 1.4:

$$w\mathbf{x} + b = 0 \tag{1.4}$$

Vectores de Soporte
Estos vectores son los puntos de la base de datos que están lo más cerca del hiperplano.

Los vectores de soporte tienen la particularidad de que son los puntos más difíciles de clasificar. Como estos puntos se encuentran en la frontera, entonces no se puede asegurar la clase a la que verdaderamente pertenecen.

Igualmente, los vectores de soporte son importantes ya que influyen en la forma de la superficie que define al hiperplano. Solo basta con mover a uno de los vectores de soporte para que cambie la forma del hiperplano.

1.3.2.4. Validación Cruzada

Los modelos de aprendizaje de máquinas tienen parámetros que deben ser optimizados, asimismo es necesario validar la estabilidad de estos modelos para saber qué tan bien se generalizarán al probar con nuevos datos. Una de las técnicas más importantes que ayudan a realizar esta validación se denomina validación cruzada (*k-fold cross-validation*) [27].

El método de *k-fold cross-validation* que se indica en la Figura 1.13 en primer lugar divide a los datos en k partes iguales. Una porción de esta división será para el entrenamiento y la restante para test.

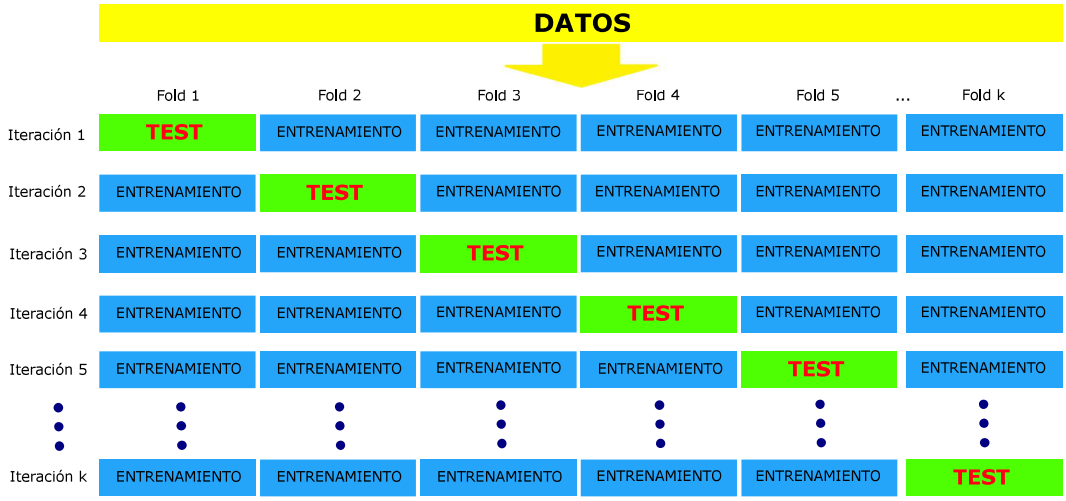


Figura 1.13: Partición de datos con *k-fold cross-validation*.

Por lo tanto, se ejecuta la iteración 1 del modelo y en este primer conjunto (*set*) de datos se obtiene un desempeño. En la siguiente iteración se corre nuevamente el modelo, pero con un set de datos diferente (como se puede observar en la Figura 1.13, los datos de test y entrenamiento son distintos en las iteraciones 1 y 2) y se alcanza un otro desempeño. Así, se continúa con este procedimiento, evaluando el desempeño del modelo con distintos *sets* de datos hasta llegar a la iteración k.

Para terminar, la medida final de desempeño del modelo, será el promedio de los valores de desempeño de cada iteración.

1.3.3. MÉTRICAS PARA LA EVALUACIÓN DEL RENDIMIENTO DE UN CLASIFICADOR

1.3.3.1. Curva ROC

La curva de Características de Funcionamiento del Receptor (*Receiver Operating Characteristics*, ROC) es una técnica para visualizar, organizar y seleccionar clasificadores en función su rendimiento [28]. Usando la curva ROC, se puede interpretar gráficamente el rendimiento de un clasificador binario variando el umbral de decisión, este umbral se refiere a un valor de corte donde todos los valores positivos que están por encima del umbral serán Verdaderos Positivos (*True Positive*, TP) y los valores negativos por encima del umbral se consideran como Falsos Positivos (*False Positive*, FP) porque son negativos que se predicen erróneamente como positivos. Ahora bien, todos los valores negativos ubicados por debajo del umbral serán Verdaderos Negativos (*True Negative*, TN) y los valores positivos por debajo del umbral serán Falsos Negativos (*False Negative*, FN) ya que son positivos que se pronostican incorrectamente como negativos.

En general, la curva ROC puede formarse con la Tasa de Verdaderos Positivos (*True Positive Rate*, TPR) versus la Tasa de Falsos Positivos *False Positive Rate*, FPR), tal como se puede ver en la Figura 1.14).

- Tasa de Verdaderos Positivos: Se conoce como la sensibilidad (*recall*) de un clasifica-

dor e indica cuántos valores positivos se clasificaron correctamente entre el total de valores positivos disponibles, esta relación se muestra en la Ecuación 1.5.

$$TPR = \text{Sensibilidad} = \frac{TP}{TP + FN} \tag{1.5}$$

- Tasa de Falsos Positivos: Es igual a $1-$ Especificidad, lo que quiere decir la relación entre los negativos erróneamente clasificados para el total de valores negativos disponibles, como se puede ver en la Ecuación 1.6.

$$FPR = 1 - \text{Especificidad} = \frac{FP}{TN + FP} \tag{1.6}$$

Cada punto que define la curva ROC es un posible candidato para ser el punto operacional del clasificador y este punto normalmente se elige para reducir la probabilidad de error, es decir, para obtener un clasificador con la mayor precisión. Sin embargo, si se quiere evaluar toda la curva ROC, entonces se usa un parámetro llamado Área bajo la curva ROC (*Area Under the ROC Curve*, AUC).

1.3.3.2. Área Bajo la Curva ROC (AUC)

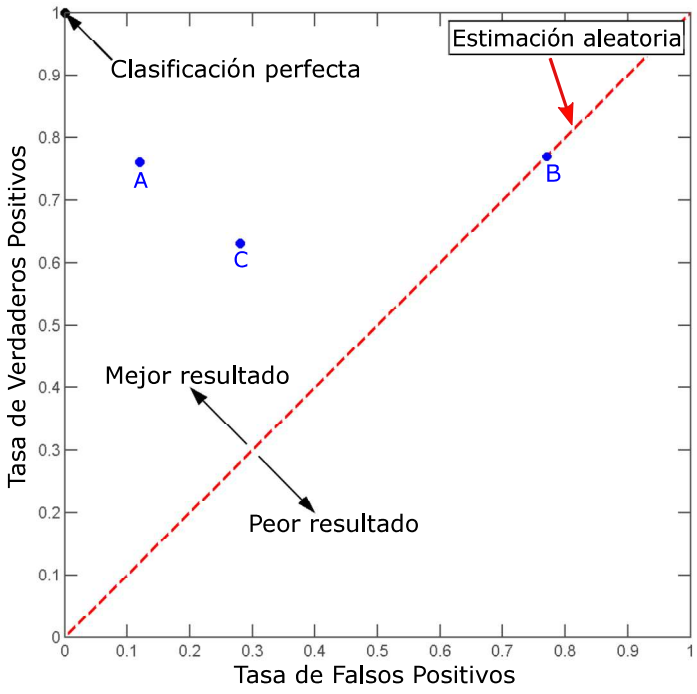


Figura 1.14: Curva ROC y AUC. Adaptado de [28].

El AUC [29] corresponde a una parte del área total de una unidad cuadrada, por lo tanto, su valor debe ser un número entre 0 y 1. A pesar de esto, específicamente para clasificadores reales, el AUC no debe ser menor que 0.5 ya que una estimación aleatoria es la que genera un área de 0.5 (ver Figura 1.14, el punto B se encuentra en la diagonal de la estimación aleatoria). El área que produce esta estimación indica que el clasificador no tiene la capacidad

de discriminar entre la clase positiva y la clase negativa. Por consiguiente, las áreas menores que 0.5 indican los peores resultados y para el caso de que el AUC sea 0, esto quiere decir que el clasificador predice la clase positiva como una clase negativa y viceversa.

Un AUC de valor 1 es un caso ideal, una clasificación perfecta que significaría que el clasificador es perfectamente capaz de distinguir entre la clase positiva y la clase negativa. Por otro lado, una curva que pase por el punto A y una curva que pase por C de la Figura 1.14 producirían áreas AUC que son aceptables.

De hecho, el AUC indica la probabilidad de que el clasificador puntúe más alto una instancia positiva elegida al azar que una negativa. Esta área se puede calcular usando un promedio de una serie de enfoques trapezoidales, i.e mediante una integración trapezoidal [29].

1.3.3.3. Incrustación de Vecinos Estocásticos Distribuidos en t (t-SNE) para visualización

La Incrustación de Vecinos Estocásticos Distribuidos en t (t-SNE) es una técnica de visualización que ayuda a reducir la dimensionalidad de los datos para poder mapear sus características de alta dimensión en características de baja dimensión. Esta técnica se aplica ampliamente en el procesamiento de imágenes, procesamiento de voz, entre otros.

De forma resumida, el funcionamiento de t-SNE es el siguiente:

- Se inicia calculando la probabilidad de similitud de los datos de alta dimensión y la probabilidad de similitud de los datos en baja dimensión respectivamente.
- Después se intenta minimizar la diferencia que hay entre estas probabilidades de similitud, tanto para el espacio de dimensiones superiores e inferiores (alta y baja dimensión). Esto se realiza con el fin de representar a los datos en el espacio de baja dimensión, de la mejor forma posible.
- t-SNE utiliza un método llamado descenso de gradiente para medir la minimización de la diferencia de las probabilidades de similitud [30].

Dicho de otra manera, la Incrustación de Vecinos Estocásticos Distribuidos en t (t-SNE) asigna los datos multidimensionales a un espacio dimensional más bajo y trata de encontrar patrones en los datos a través de la identificación de la similitud de los datos con múltiples características.

El proceso de t-SNE hace que las características de entrada ya no sean identificables, por lo que no se puede hacer ninguna inferencia solamente basado en el resultado de la salida de t-SNE. Como consecuencia, esta técnica se usa principalmente para exploración y visualización de datos.

En la Figura 1.15, se muestra un ejemplo del resultado de t-SNE para visualizar una base de datos de dígitos cuya dimensión original es 64. En este caso, se visualizan los datos reducidos su dimensionalidad para un espacio de dimensión 2D.

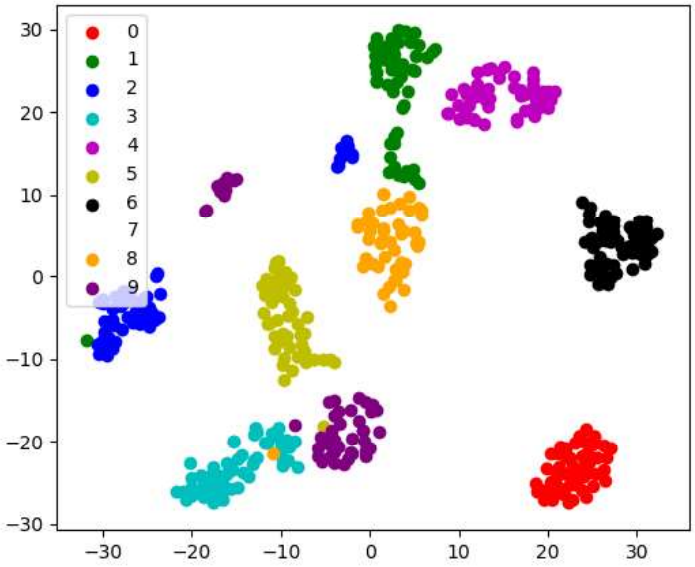


Figura 1.15: Ejemplo de la visualización de datos con t-SNE. Fuente [31].

1.3.4. TRABAJOS ANTERIORES

En la literatura científica, a pesar de que algunas personas han propuesto varias formas de medir la calidad de los granos de cacao, hasta donde se sabe, no hay trabajos previos centrados específicamente en los granos de cacao con pulpa. Por ejemplo, en [32], desarrollaron una aplicación para teléfonos Android para medir automáticamente el nivel de infección de las mazorcas de cacao sin retirarlas de la planta y además bajo diferentes condiciones de luz.

En otros trabajos [33] y [34], también procesan imágenes de mazorcas de cacao, donde el objetivo principal del primero es determinar qué tan maduras están estas mazorcas utilizando Redes Neuronales Artificiales (*Artificial Neural Networks*, ANN). Mientras que, para el segundo, su objetivo es medir el nivel de infección de las mazorcas de cacao mediante un clasificador SVM.

Por otro lado, los trabajos [35–39] proponen estimar el grado de fermentación de los granos de cacao sin pulpa, para lo cual se estudian diferentes técnicas de segmentación como Otsu en [35] y [37], k-means en [35], entre otros. En cuanto a la etapa de clasificación, [36] y [37] utilizaron SVM, [35] empleó el Aprendizaje de Cuantificación Vectorial (*Learning Vector Quantization*, LVQ), [38] utilizó Redes Neuronales Artificiales (*Artificial Neural Networks*, ANN) y [39] aplicó el Sistema Adaptativo de Inferencia Neuronal Difuso (*Adaptive Neural-Fuzzy Inference System*, ANFIS).

A diferencia de los estudios mencionados anteriormente, el presente trabajo determina la calidad de los granos de cacao a partir de imágenes de los mismos fuera de la mazorca, i.e., los granos de cacao frescos con pulpa (ver Figura 1.1).

2. METODOLOGÍA

En esta sección se describirá la propuesta para clasificar granos de cacao frescos basados en visión computacional, para lo cual primero partiremos con el procedimiento para remover el fondo de las imágenes usando el algoritmo k-means que se detallará en la Sección 2.1. Luego, en la Sección 2.2 se describirán los pasos para extraer las características de la pulpa mediante un enfoque de Bolsa de Palabras Visuales (BoVW).

Finalmente, en la Sección 2.3 se mencionará la utilización de SVM y la validación cruzada (*K-fold cross-validation*) en la clasificación de los granos de cacao en excelente y mala calidad. La Figura 2.1 muestra un diagrama de bloques de la propuesta planteada.

Figura 2.1: Diagrama de bloques del clasificador de imágenes de granos de cacao frescos propuesto.

Para este trabajo se recopiló una base de datos de 247 imágenes de granos de cacao frescos (con pulpa). Estas imágenes se clasificaron previamente en excelente calidad (125 imágenes) y mala calidad (122 imágenes) de acuerdo al conocimiento sensorial de los productores experimentados de cacao de Los Ríos, Ecuador.

Las Figuras 2.2.a, 2.2.b, 2.2.c, indican imágenes de granos de cacao frescos de EXCELENTE CALIDAD y en las Figuras 2.2.d, 2.2.e, 2.2.f se tienen imágenes de granos de cacao frescos de MALA CALIDAD de la base de datos recopilada. Aunque se intentó mantener el fondo lo más uniforme y contrastante posible para facilitar su remoción, se puede observar que la humedad natural de la pulpa y otros artefactos hacen que el fondo no sea uniforme

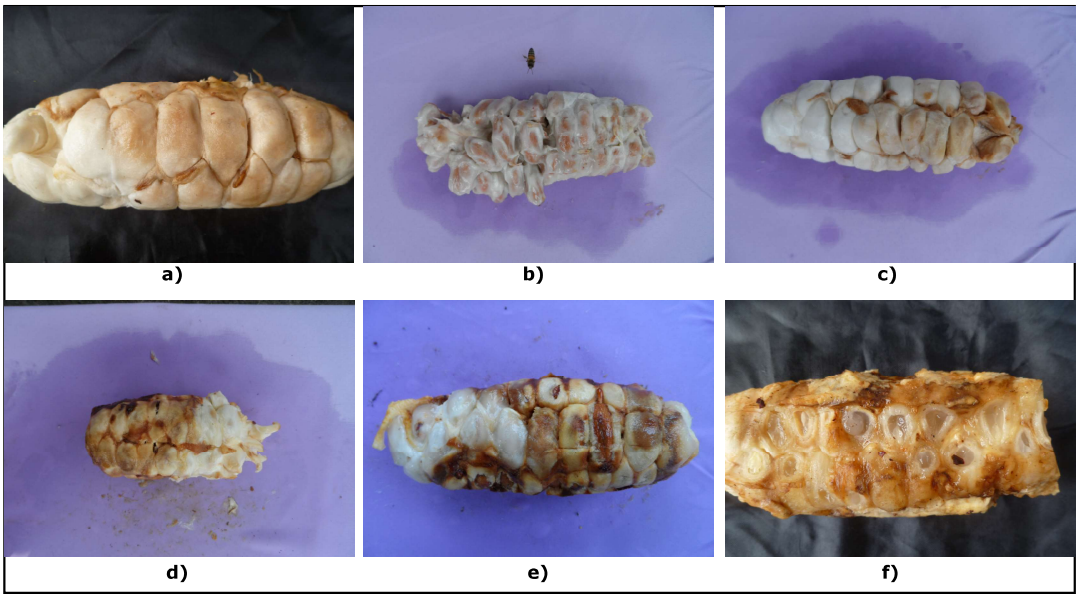


Figura 2.2: Ejemplos de imágenes de EXCELENTE CALIDAD y MALA CALIDAD de la base de datos recopilada.

afectando su contraste con respecto a la fruta.

2.1. REMOCIÓN DEL FONDO USANDO K-MEANS

En esta sección se explica a detalle como remover el fondo de las imágenes de la base de datos de los granos de cacao frescos, donde fundamentalmente su procedimiento se indica en la Figura 2.3.

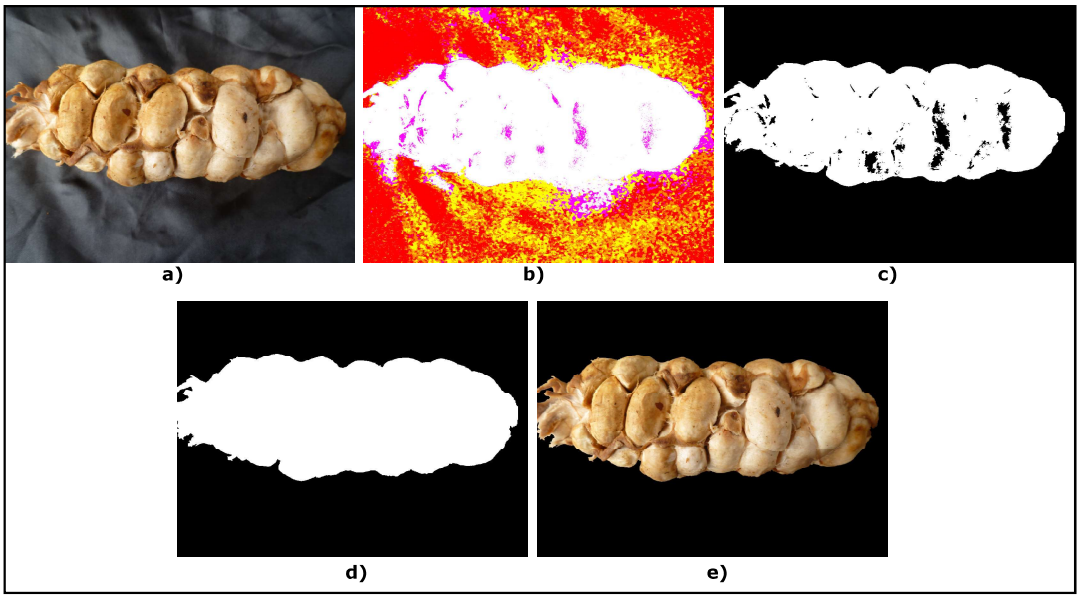


Figura 2.3: Etapas de procesamiento utilizadas para remover el fondo de las imágenes de cacao.

Primero, en la Figura 2.3.a se redimensiona a las imágenes de 4320×3240 píxeles a $1080 \times$

810 píxeles. Esto se realizó ya que el tamaño original de las imágenes era grande (aproximadamente 14 megapíxeles cada una) y demandaba altos requisitos del procesador para ejecutar el programa.

Entonces, mediante el uso del comando `imresize` que se indica en el Segmento de código 2.1, se disminuyó el tamaño de las imágenes a ser procesadas.

Segmento de código 2.1: Uso del comando imresize

```
1       % Se redimensiona la imagen a un 25 %
2       img_ori1 = imresize(img_ori1, 0.25);
```

El comando `imresize` utilizado tiene los siguientes argumentos: `imresize(img_ori1, scale)`

- `img_ori1`: Hace referencia a la imagen que se va a cambiar de tamaño. La imagen debe estar especificada como una matriz numérica real y no dispersa.

- `0.25`: Representa el factor de cambio de tamaño, es un valor real, un escalar numérico.

  En esta parte se configuró un valor de 0.25 el cual indica que la imagen disminuirá su tamaño a 25 %.

Con `imresize` pasamos de nuestras imágenes originales (ver Figura 2.4.a) a las imágenes redimensionadas (ver Figura 2.4.b), tal como se puede observar en el ejemplo de la Figura 2.4.

Luego, en la Figura 2.3.b se realiza una conversión del espacio RGB al espacio CIE 1976 L*a*b* [40] para definir los colores independientemente de cómo se producen o se muestran.

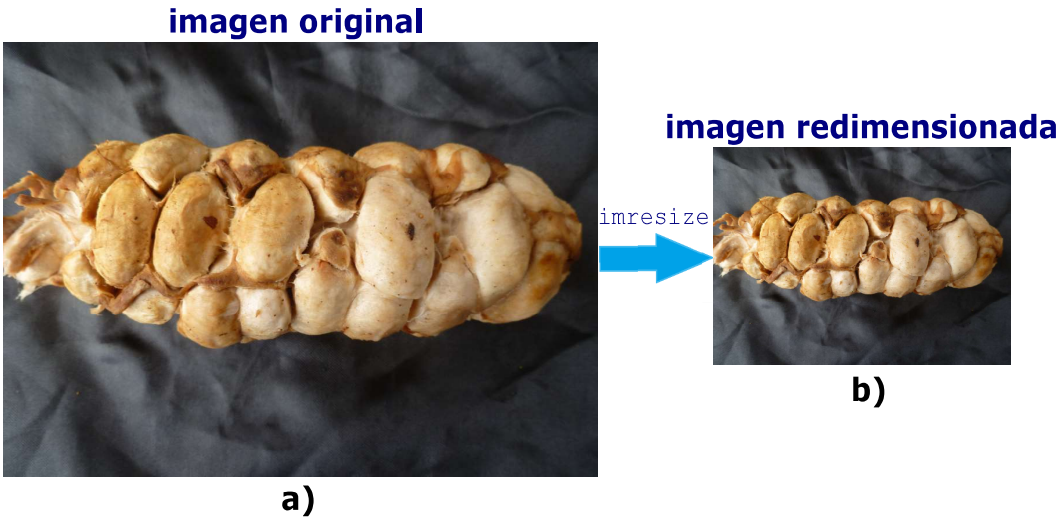


Figura 2.4: Ejemplo del resultado del comando `imresize` para un factor de cambio de 0.25.

A continuación, en el Segmento de código 2.2, se indica el comando `rgb2lab` y el comando `im2single` empleados para efectuar la conversión primero al espacio CIE 1976 L*a*b* y posteriormente la transformación de estos datos a un tipo de datos single, los cuales se usarán con el comando `imsegkmeans`.

Segmento de código 2.2: Uso de los comandos rgb2lab e im2single

```
% Se convierte la imagen de color (RGB) al espacio CIE 1976 L*a*b*
labsp_cco = rgb2lab(cco);
ab_color = labsp_cco(:,:,2:3);
% Se convierten los parámetros a* y b* a un tipo de datos sigle
ab_sin = im2single(ab_color);
```

- Los argumentos del comando `rgb2lab(cco)` son los siguientes:

  1. `cco`: Es el argumento de entrada que representa a una matriz numérica que contiene los valores de las intensidades de color Rojo, Verde y Azul.

  2. `labsp_cco`: Esta variable es el argumento de salida que contiene los valores de color L*, a* y b* convertidos a través del comando `rgb2lab(cco)`. Igualmente, `labsp_cco` es una matriz numérica que tiene el mismo tamaño que el argumento de entrada pero es de tipo double.

     Con esto en mente, el resultado de aplicar el comando `rgb2lab` (ver Figura 2.5.b) sobre la imagen `cco` (ver Figura 2.5.a) es el que se muestra en la Figura 2.5.

     En nuestra variable `ab_color` extraemos la información de color existente en los canales a* y b* correspondientes a 2:3, i.e., desde el canal 2 hasta el canal 3.

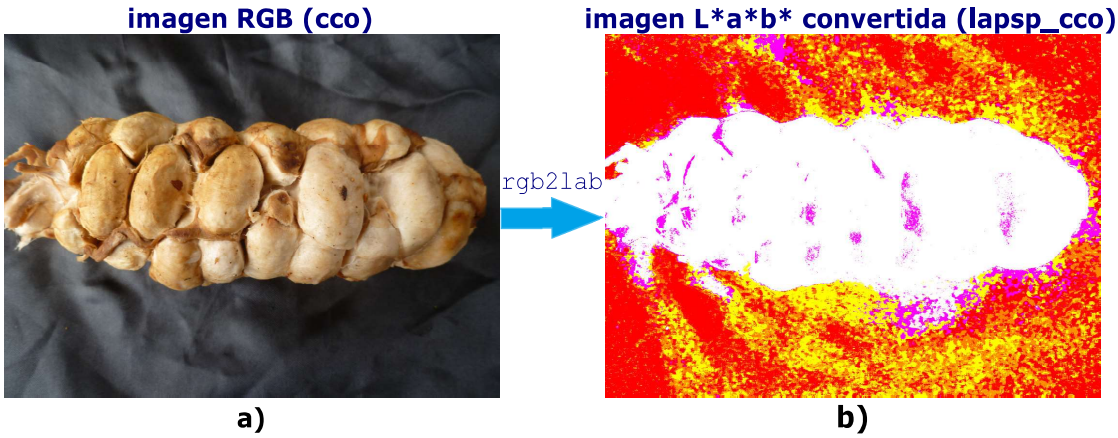


Figura 2.5: Ejemplificación del uso del comando `rgb2lab`.

- Con el comando `im2single(ab_color)`, se toma a la imagen anterior `ab_color` y se le realiza una conversión a precisión simple. Esta conversión es dependiente del tipo de imagen que se encuentra a la entrada.

  De ahí que, al ser nuestra imagen de entrada no del tipo single, en la salida del comando `im2single` se obtendrá una imagen equivalente a la clase single `ab_sin`, compensando o reescalando sus datos según sea necesario.

Después, se aplica k-means *clustering* para obtener la imagen binaria, donde la fruta es de color blanco. Entonces, se utilizó el comando `imsegkmeans`, mismo que está destinado a segmentar la imagen `ab_sin` en un número determinado de grupos, además por medio de la manipulación de los argumentos adicionales que tiene el algoritmo k-means clustering se cambió el enfoque que tiene por defecto.

A la salida el comando `imsegkmeans` se consiguen las etiquetas segmentadas correspondientes a cada clúster. Tal como se puede observar en el Segmento de código 2.3, los argumentos de entrada y la salida del comando `imsegkmeans` son:

Segmento de código 2.3: Aplicación de k-means clustering

```
% APLICACIÓN DE K-MEANS CLUSTERING
% Se repite 2 veces el proceso de clustering k-means
n_clusters = 2;
% n_clusters: Indica el números de clústers
% ab_sin: Corresponde a la imagen a segmentar
% 'NumAttempts': Es el número de veces que se repite el …
   proceso de
% agrupación
% 'NormalizeInput': Indica la normalización de los datos …
   de entrada
% 'MaxIterations': Es el número máximo de iteraciones
% 'Threshold': Indica el umbra de precisión
etiqueta_pixel = imsegkmeans(ab_sin, n_clusters, ...
    'NumAttempts', 2, 'NormalizeInput', false, ...
    'MaxIterations', 100, 'Threshold', 5);
```

- `ab_sin`: Es la imagen a segmentar y debe estar especificada ya sea como una imagen en escala de grises 2D, una imagen en color 2D o una imagen multiespectral 2D. El comando admite los siguientes formatos: single, int8, int16, uint8 y uint16. Es por esta razón que se convirtió previamente a nuestra imagen en tipo single.

- `n_clusters`: Se refiere al número de grupos o *clusters* en el que se segmentará a la imagen. Este argumento debe ser un número entero positivo.

Para nuestro propósito, se configuró el número de clusters en 2 porque uno de ellos representará al fondo y el otro a la fruta. De antemano, no se sabe si la fruta estará en el primer o en el segundo clúster.

- `'NumAttempts'`: Es un argumento que corresponde al número de veces que se va a repetir el proceso de agrupación o *clustering*. Cada repetición se efectúa utilizando una nueva posición inicial del centroide del clúster.

  Por defecto este valor es el número 3, no obstante, tras realizar varias pruebas se configuró en 2 así como se puede ver en el Segmento de código 2.3.

- `'NormalizeInput'`: Permite normalizar los datos de entrada para tener una media cero y la varianza unitaria.

  Se puede configurar como *true* (por defecto) o *false*. En este caso se digitó *false* indicando que la normalización de cada canal de entrada no será de forma individual.

- `'MaxIterations'`: Representa al número máximo de iteraciones cuyo valor predeterminado es de 100. En esta ocasión mantuvimos en la configuración el valor de 100.

  El algoritmo k-means es iterativo y en algún momento se debe detener porque de lo contrario las iteraciones se harían hasta el infinito. Por esto, hay que definir un número máximo de iteraciones (`'MaxIterations'`) y un umbral (`'Threshold'`), los cuales están relacionados entre sí y ayudan a parar al algoritmo.

- `'Threshold'`: Es el umbral de precisión que tiene por valor predeterminado a 1×10^{-4}. Este número debe ser positivo y en el programa de configuró con un valor de 5.

- `etiqueta_pixel`: Este argumento de salida es la matriz de etiquetas constituida por enteros positivos, donde los píxeles que tengan la etiqueta '1' corresponderán al primer clúster, la etiqueta '2' al segundo clúster y así sucesivamente hasta llegar al valor de `n_cluster`. Como se estableció el número de clusters en 2, por lo tanto, nuestras etiquetas serán '1' y '2'.

En la Figura 2.6 se puede observar el uso del comando `imsegkmeans` sobre `ab_sin` (de tipo single) para obtener la imagen etiquetada por índice de clúster `etiqueta_pixel`. Seguidamente, a través del Segmento de código 2.4 se separan los objetos que poseen los píxeles con la etiqueta '1', i.e. que están en el clúster 1 (ver Figura 2.6.a) y los objetos que tienen los píxeles con la etiqueta '2', i.e. localizados en el clúster 2 (ver Figura 2.6.b).

Segmento de código 2.4: Separación de los objetos en el clúster 1 y en el clúster 2

```
1       % Se asigna a mascara1 el clúster = 1, que devuelve el …
           comando imsegkmeans
```

```
2       mascara1 = etiqueta_pixel == 1;
3       cluster1 = cco .* uint8(mascara1);
4       % Se asigna a mascara21 el clúster = 2, que devuelve el …
           comando imsegkmeans
5       mascara21 = etiqueta_pixel == 2;
6       cluster2 = cco .* uint8(mascara21);
```

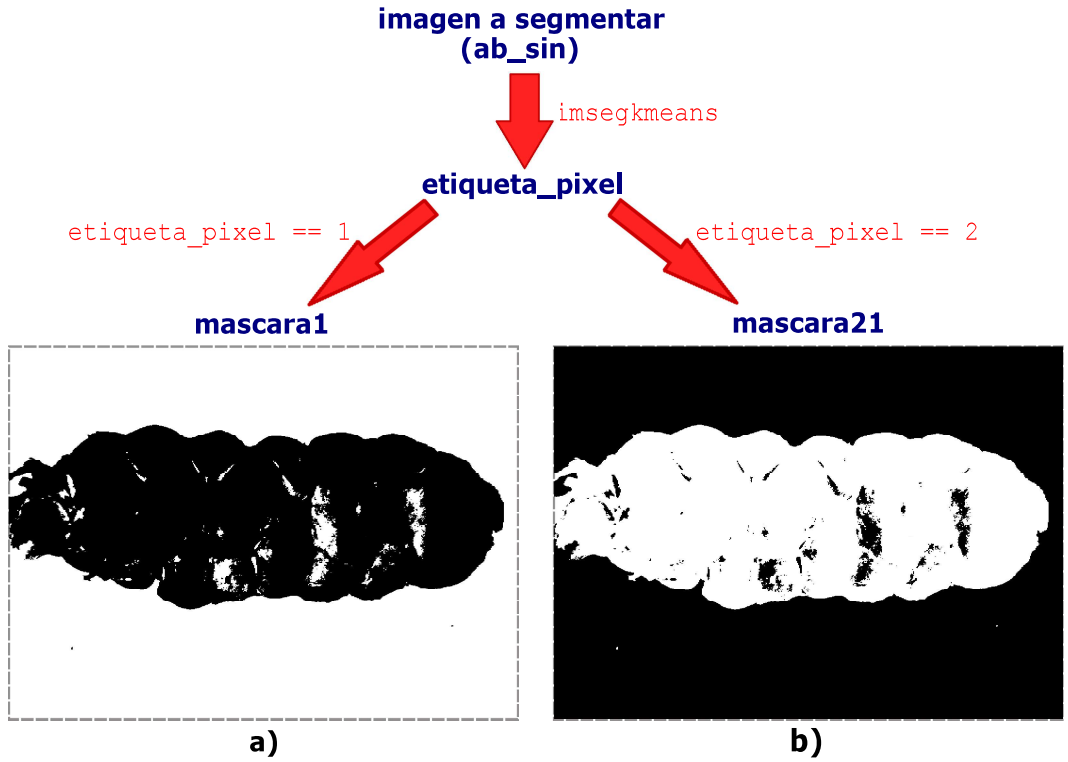


Figura 2.6: Resultado de la aplicación del comando `imsegkmeans` sobre `ab_sin`.

En el paso anterior se configuró el algoritmo k-means para devolver dos *clusters*. Sin embargo, no se sabe a priori en qué clúster está la fruta.

Por ejemplo, en la Figura 2.7.a la fruta se encuentra en el clúster 2. Por el contrario, en la Figura 2.7.b la fruta se sitúa en el clúster 1.

Por lo tanto, en primer lugar se extrae el objeto más grande de la imagen segmentada con k-means en ambos *clusters* (i.e. en las imágenes binarias `mascara1` y `mascara21`) con el comando `bwareafilt` así como se puede ver en el Segmento de código 2.5.

Segmento de código 2.5: Uso del comando bwareafilt

```
1       % Se extrae el objeto más grande de la imagen segmentada …
           para ambos clusters
2       a1 = bwareafilt(mascara1,1);
3       a2 = bwareafilt(mascara21,1);
```

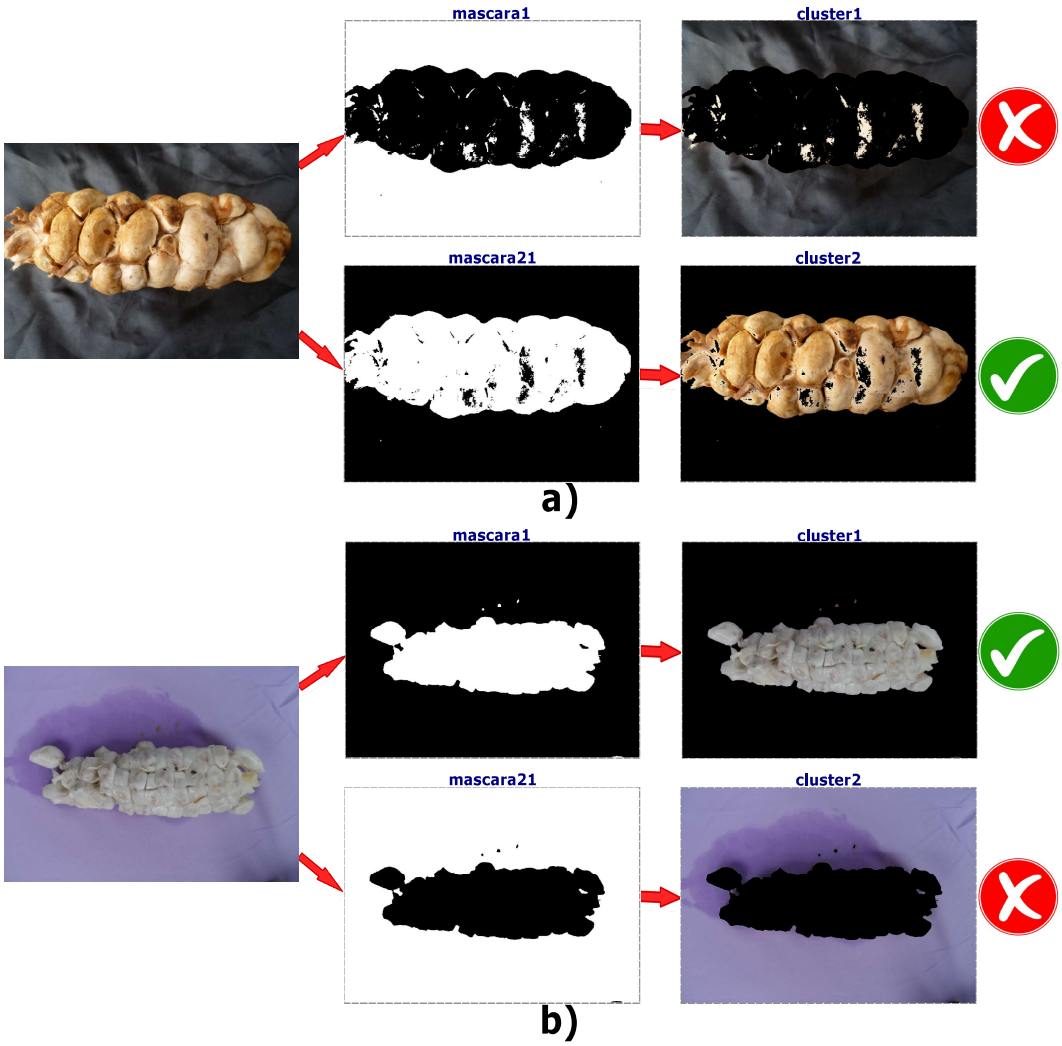


Figura 2.7: Localización del cacao en el clúster correspondiente.

Donde `bwareafilt` se compone de los siguientes argumentos:

- `mascara1`: Es la imagen binaria (de tipo logical) que se filtrará para obtener de ella un número determinado de objetos que sean los más grandes.
- `1`: Indica el número de objetos, los más grandes, que se extraerán de la imagen. En nuestra configuración colocamos el valor de '1' porque nos interesa solo obtener el objeto más grande de cada clúster.
- A la salida en la variable `a1` nos devuelve la imagen binaria filtrada.

Las imágenes binarias resultantes `a1` y `a2` tras aplicar el comando `bwareafilt` sobre `mascara1` y `mascara21` se indican en la Figura 2.8.a y en la Figura 2.8.b respectivamente.

Se puede observar en la Figura 2.8 que `bwareafilt` efectivamente extrae el objeto más grande en color blanco. De manera que, por ejemplo los objetos encerrados en círculo rojo

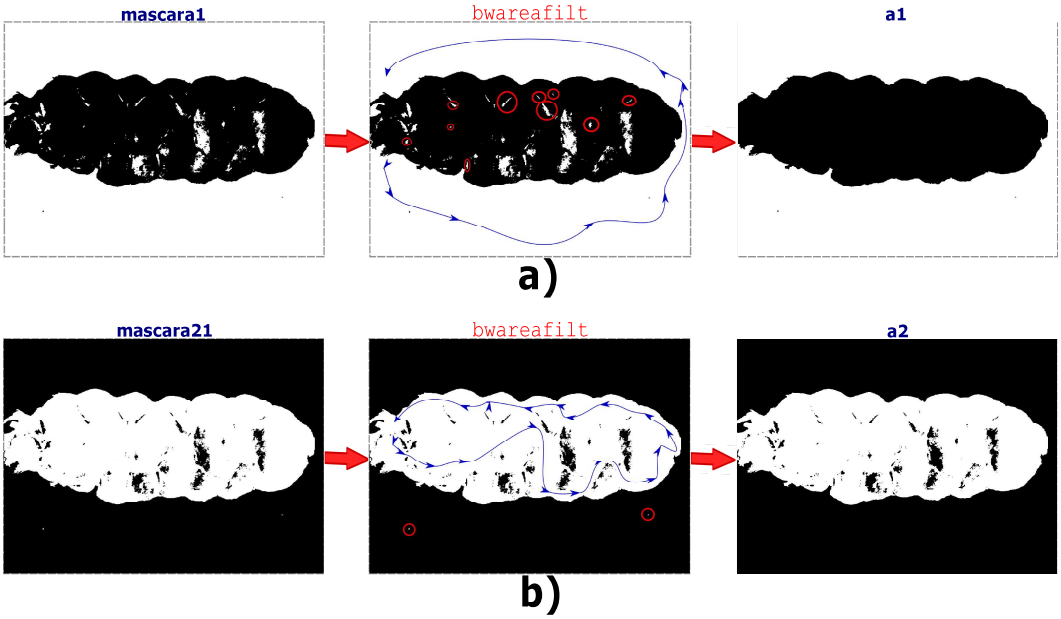


Figura 2.8: Ejemplificación del uso del comando `bwareafilt`.

desaparecen en las imágenes `a1` y `a2` quedando solo el objeto más grande representado por la línea azul con flechas.

Es así que, se termina con dos objetos que son candidatos admisibles. Por consiguiente, para discriminar cuál es la fruta, se calcula el área de ambos objetos candidatos (`a1` y `a2`) y se elige aquel cuya área es la más pequeña, tal como se puede ver en la Figura 2.3.c.

En el Segmento de código 2.6, se aplicó el comando `regionprops` sobre las imágenes binarias de entrada `a1` y `a2` que junto con la configuración de sus argumentos de entrada se obtienen ciertas medidas de la forma de estas imágenes.

Segmento de código 2.6: Uso del comando regionprops y cálculo de las áreas de los objetos

```
1    % Se obtiene las dimensiones del rectángulo que contiene …
       al objeto extraído
2    Bmask1 = regionprops(a1,'BoundingBox');
3    % Se determina el área del rectángulo que contiene al objeto
4    Amask1 = Bmask1.BoundingBox(3)*Bmask1.BoundingBox(4);
5    Bmask21 = regionprops(a2,'BoundingBox');
6    Amask21 = Bmask21.BoundingBox(3)*Bmask21.BoundingBox(4);
```

- `'BoundingBox'`:, es el argumento del comando `regionprops` que representa una propiedad específica la cual permite adquirir las dimensiones del rectángulo que encierra a nuestro objeto de interés, i.e. la región consistente de píxeles blancos.
- En la variable de salida `Bmask1` se devuelve una matriz de estructuras donde sus elementos son los siguientes: $[x, y, ancho, altura]$.

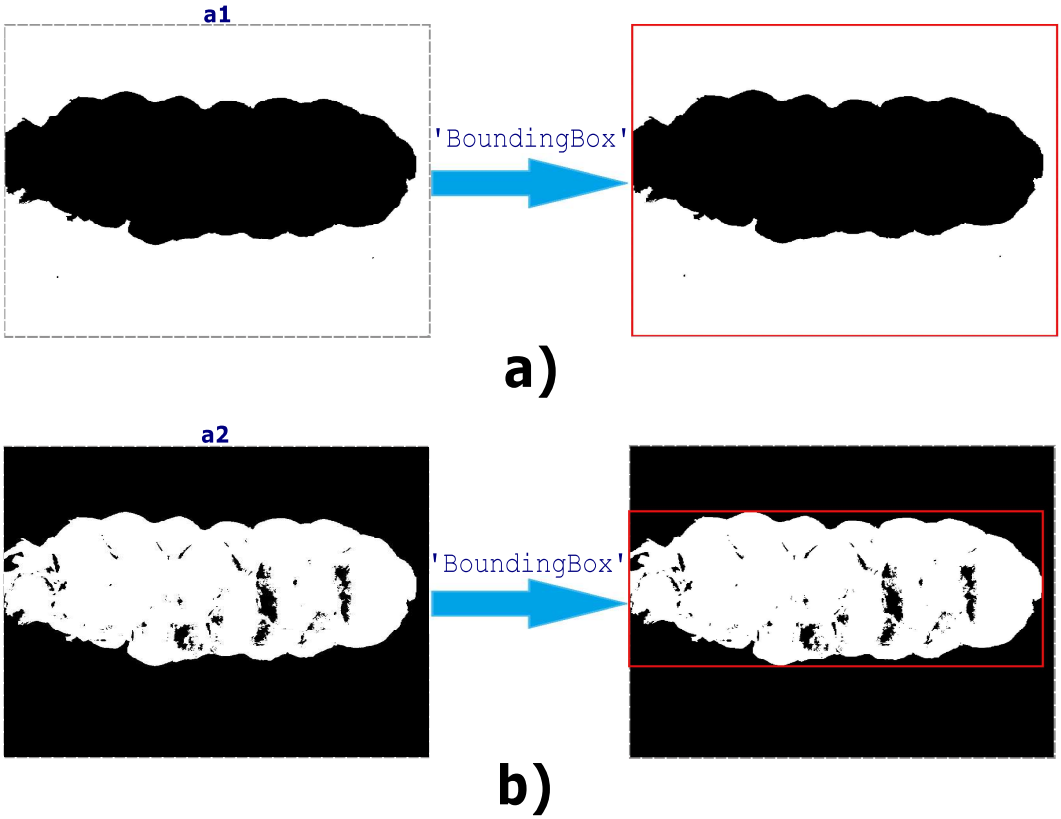


Figura 2.9: Ejemplo de cómo actúa el argumento `'BoundingBox'` en el comando `regionprops`.

- Para calcular el área que contiene a nuestros objetos `a1` y `a2` usamos los valores de los elementos que están en las posiciones 3 y 4 que son ancho y altura respectivamente de las estructuras antes mencionadas. Por tanto, las áreas serán la multiplicación de estos valores.

En la Figura 2.9, se puede observar a manera de ejemplificación el efecto del argumento `'BoundingBox'` al encerrar en un rectángulo rojo al objeto con píxeles blancos en la imagen `a1` (ver Figura 2.9.a) y en la imagen `a2` (ver Figura 2.9.b).

Antes que nada, con las áreas determinadas, se realiza la comparación entre las mismas para establecer la máscara que nos ayudará a remover el fondo de las imágenes, tal como se muestra en el Segmento de código 2.7.

Segmento de código 2.7: Condiciones para determinar la máscara de remoción de fondo

```
1      % ESTABLECIMIENTO DE LAS CONDICIONES PARA DETERMINAR LA ...
          MÁSCARA
2      % Se compara las áreas
3      if (Amask1<Amask21)
4          Bin_W2=a1;
5      else
```

```
6           Bin_W2=a2;
7       end
```

- En el desarrollo de esta parte del código se verifica que la menor área (perteneciente a los píxeles blancos) entre `Amask1` y `Amask21` corresponde al cacao y su imagen binaria representa a la máscara correcta.

Finalmente, en la Figura 2.3.d se llena los huecos usando una operación morfológica y se emplea esta imagen como una máscara para aislar la fruta, como en la Figura 2.3.e.

La operación morfológica empleada se ejecutó a través del comando `imfill` que se indica en el Segmento de código 2.8 y cuya función es la siguiente:

Segmento de código 2.8: Uso del comando imfill

```
1       % Se llena los huecos de la imagen binaria
2       mascara = imfill(Bin_W2,'holes');
3       % Se multiplica la imagen con la máscara obtenida, para …
           tener la imagen removida el fondo
4       cluster_rem = cco .* uint8(mascara);
5       c = cluster_rem;
```

- El comando `imfill` trabaja en base a un algoritmo de reconstrucción morfológica, en esta parte al colocar el argumento `'holes'`, se está llenando los agujeros de nuestra imagen binaria `Bin_W2`.

- Cabe mencionar que un hueco será considerado al conjunto de píxeles del fondo que no se pueden alcanzar llenando el fondo desde un borde de la imagen.

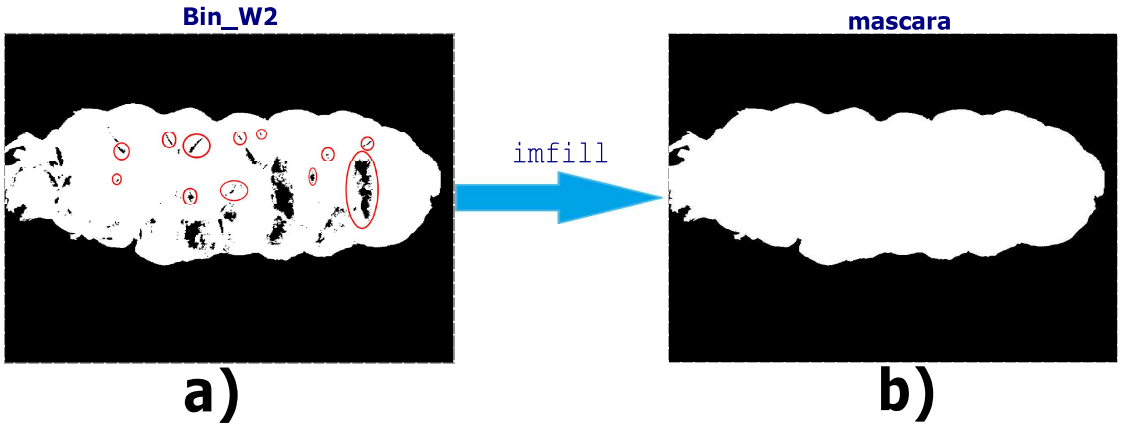


Figura 2.10: Resultado de aplicar la operación morfológica `imfill` sobre la imagen `Bin_W2`.

Tal como se puede observar en la Figura 2.10, lo que está encerrado en círculos rojos son los huecos que se quiere llenar y aplicando `imfill` a la imagen binaria `Bin_W2`, pasamos de la Figura 2.10.a a la Figura 2.10.b.

Tras haber seguido todos los pasos anteriores, es preciso señalar que en las Secciones 2.2 y 2.3 se trabajará con las imágenes removidas el fondo, las cuales están en la base de datos denominada `imds2`.

2.2. EXTRACCIÓN DE CARACTERÍSTICAS

Para la extracción de características, se utilizó el modelo de Bolsa de Palabras Visuales (BoVW) que se puede observar en la Figura 2.11 y que consta de los siguientes pasos [8]:

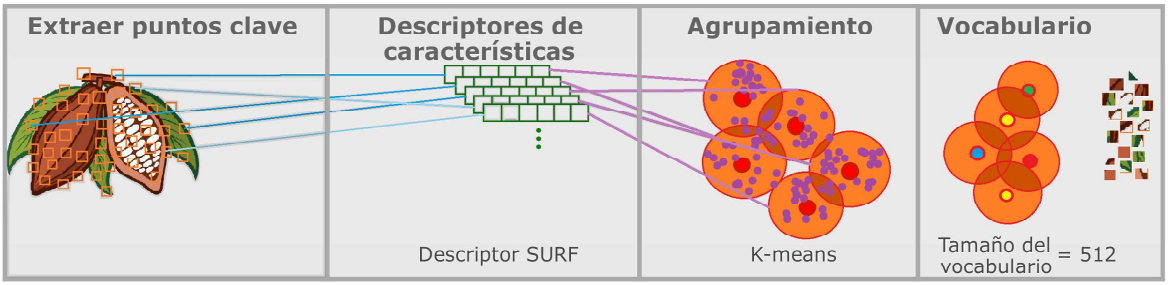


Figura 2.11: Representación del proceso de flujo de trabajo del algoritmo Bolsa de Palabras Visuales con los datos del presente trabajo.

1. Extraer los puntos clave o también llamados puntos de interés. En general, hay dos formas de llevar a cabo este paso, ya sea utilizando una cuadrícula o un detector de puntos.

   Aquí se usó el detector de puntos de Características Robustas Aceleradas (*Speeded Up Robust Features*, SURF) [15] principalmente porque esta técnica es más rápida con respecto al algoritmo de Transformada de Características Invariantes a la Escala (*Scale Invariant Feature Transform*, SIFT) [18].

2. Luego, para cada punto de interés detectado en el paso anterior, el algoritmo analiza su vecindad y extrae de él un descriptor de características. Específicamente, se empleó el descriptor SURF de 64 dimensiones.

3. Posteriormente, sobre los descriptores de características, aplicamos el algoritmo k-means como método de agrupamiento para obtener 512 *clusters*. Con el proceso de agrupamiento (*clustering*) juntamos a los descriptores que poseen características similares.

4. Después de que se tienen estos grupos (*clusters*), se procede a determinar los centroides de cada uno de ellos. En la Figura 2.12 se puede ver que nuestro número de centroides es de 512. Como resultado, las palabras visuales de nuestro vocabulario son los centroides de esos *clusters*.

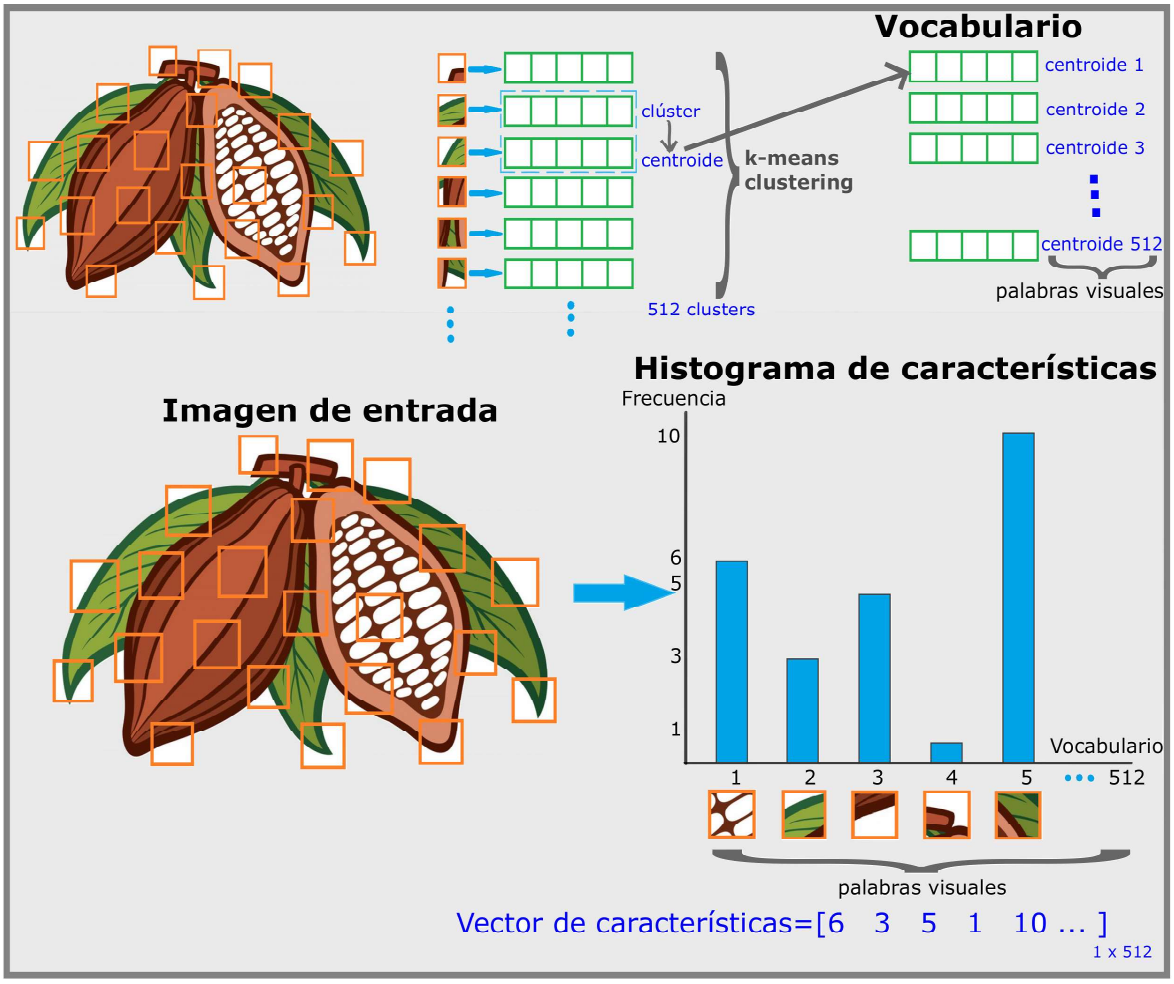


Figura 2.12: Ejemplo del histograma de características de una imagen de entrada y su vector de características correspondiente con un vocabulario de 512 palabras visuales.

5. Por otra parte, una vez que tenemos el vocabulario, para cada imagen se construye un histograma a partir de la frecuencia de las palabras visuales en la imagen, como se puede ver en la Figura 2.12. Los histogramas que se generan de las imágenes de entrada se representan en vectores de características que proporcionaremos al clasificador en la Sección 2.3.

A continuación, en el Segmento de código 2.9 se especifica el uso y los valores que se establecieron en los argumentos del objeto `bagOfFeatures`.

Segmento de código 2.9: Aplicación y uso del comando bagOfFeatures

```
1    Bolsa1 = bagOfFeatures (imds2_entrenamiento_fold, ...
2        'VocabularySize', 512, 'StrongestFeatures', 0.8, ...
3        'PointSelection', 'Detector');
```

- `imds2_entrenamiento_fold`: Son las imágenes de entrenamiento precisadas como un objeto *ImageDatastore*.

  Particularmente, un objeto *ImageDatastore* permite administrar un conjunto de imágenes, donde cada una de ellas crea un espacio en memoria, pero el conjunto total de estas imágenes no necesariamente lo hace.

- `'VocabularySize'`: Este argumento permite definir la propiedad Número de Palabras Visuales del objeto `bagOfFeatures`. En esta parte se eligió el valor de 512 ya que se obtuvo una mejor exactitud y se demoraba menos tiempo en comparación, por ejemplo con el valor de 2048.

  Del mismo modo, se puede observar en la Figura 2.12 que el número de *clusters* es igual a nuestro tamaño de vocabulario que se estableció a través del argumento `'VocabularySize'` del comando `bagOfFeatures`.

- `'StrongestFeatures'`: Mediante esta propiedad se pude configurar un valor numérico que esté en el rango de $[0, 1]$ para indicar la fracción de las características más fuertes de cada etiqueta de `imds2_entrenamiento_fold`. En el Segmento de código 2.9 se puede ver que en esta propiedad se dispuso el valor de 0.8 que también corresponde al valor por defecto del comando.

- `'PointSelection'`: Se refiere a un método de selección de las ubicaciones de los puntos de interés (detector o cuadrícula) en la extracción de características de la imagen a través de la técnica de Características Robustas Aceleradas (SURF), como se mencionó y observó en el paso 1 de la Figura 2.11.

  En este trabajo, se estableció el `'PointSelection'` en `'Detector'`, lo que quiere decir que se fijó al algoritmo SURF para trabajar en la función de detector de los puntos clave.

- `Bolsa1`: Este argumento de salida representa a nuestra Bolsa de Palabras Visuales construida y se especifica como un objeto *bagOfFeatures* el cual puede utilizarse en la clasificación de las categorías de imágenes.

2.3. CLASIFICACIÓN

Ahora bien, en esta parte se desarrollaron dos funciones en MATLAB que son llamadas desde el programa principal `main_cocoa_classifier.m` (ver ANEXO A): la primera función es `bovwfold_linear.m` y la segunda `bovwfold_gaussian.m` (ver ANEXO B y ANEXO C). En ambas funciones se efectúan tanto la clasificación como el proceso de validación cruzada (*K-fold cross-validation*), pero se diferencian en que la primera usa en la clasificación la función de núcleo lineal (*linear kernel function*), mientras que la segunda usa la función de núcleo gaussiano (*gaussian kernel function*).

Así pues, con lo que respecta a la clasificación, se empleó el modelo de Máquinas de Vectores de Soporte (SVM). De hecho, SVM al ser un clasificador binario de algoritmo supervisado fue de gran ayuda para este trabajo en el que se tenía dos clases: EXCELENTE CALIDAD y MALA CALIDAD. Además, el algoritmo SVM se lo puede usar tanto en problemas de clasificación como de regresión y particularmente se utilizó el algoritmo para el problema de clasificación.

En el Segmento de código 2.10 y en el Segmento de código 2.11 se pueden apreciar las configuraciones del clasificador SVM que se usaron en este trabajo para establecer las funciones de núcleo lineal (*linear kernel function*) y de núcleo gaussiano (*gaussian kernel function*) respectivamente. Esto se realizó con el objetivo de establecer qué función de núcleo del algoritmo SVM nos permitía lograr la mejor clasificación de nuestros datos.

Segmento de código 2.10: Aplicación del clasificador SVM lineal

```
% CLASIFICADOR SVM LINEAL
imds2_entrenamiento = imds2_entrenamiento_fold.Labels;
 clasificacion_SVM = fitcsvm(...
 Mat_caract_entrenamiento, ...
 imds2_entrenamiento, ...
 'KernelFunction', 'linear', ...                                    % ...
    Se especifica la función kernel como lineal
 'OptimizeHyperparameters', 'auto', ...
 'Standardize', true, ...
 'ClassNames', categorical({'EXCELENTE CALIDAD'; 'MALA ...
    CALIDAD'}));
```

Segmento de código 2.11: Aplicación del clasificador SVM gaussiano

```
% CLASIFICADOR SVM GAUSSIANO
imds2_entrenamientog = imds2_entrenamiento_fold.Labels;
 clasificacion_SVMg = fitcsvm(...
 Mat_caract_entrenamiento, ...
 imds2_entrenamientog, ...
 'KernelFunction', 'gaussian', ...                                  % ...
    Se especifica la función kernel como gaussiana
 'OptimizeHyperparameters', 'auto', ...
 'Standardize', true, ...
 'ClassNames', categorical({'EXCELENTE CALIDAD'; 'MALA ...
    CALIDAD'}));
```

Como resultado, se escogió al clasificador SVM con la función de *kernel* lineal, en virtud de que este *kernel* es más fácil de interpretar, por su baja complejidad computacional y debido

a que es más rápido que la función de kernel gaussiano. Cabe señalar que se han seleccionado los mejores valores para el hiperparámetro utilizando un método de optimización Bayesiano [20]. Seguidamente, del Segmento de código 2.10 se presentan los argumentos fijados para el comando `fitcsvm` en la aplicación del clasificador SVM lineal.

Donde el comando `fitcsvm` es el que permite entrenar un clasificador SVM para la clasificación de dos clases de datos de acuerdo a las especificaciones de los siguientes argumentos:

- `Mat_caract_entrenamiento`: Representa a la matriz de características de los datos de entrenamiento que se obtiene al codificar las imágenes de entrenamiento con la Bolsa de palabras visuales `Bolsa1` (ver Segmento de código 2.9).

- `imds2_entrenamiento`: Son las etiquetas de clase de las imágenes de entrenamiento representadas en un vector. Estas etiquetas se consiguen colocando .Labels como sufijo de las imágenes de entrenamiento `imds2_entrenamiento_fold` (ver Segmento de código 2.9).

- `'KernelFunction'`: Este argumento se refiere a la función de núcleo que puede ser establecida como: `'gaussian'`, `'linear'` ó `'polynomial'`. En este caso, se seleccionó `'linear'` lo que indica una función de núcleo o kernel lineal predeterminado para el aprendizaje binario.

- `'OptimizeHyperparameters'`: Corresponde a la optimización de hiperparámetros donde el argumento por defecto es `'none'` que significa no optimizar. Sin embargo, en este proyecto se utilizó el argumento `'auto'` que apunta a realizar una optimización bayesiana de hiperparámetros.

- `'Standardize'`: Representa al marcador o bandera para estandarizar los datos del predictor `Mat_caract_entrenamiento` y tiene como valor predeterminado a `false`. No obstante, en esta parte se configuró este valor en `true` indicando que MATLAB entrenará al clasificador aplicando predictores estandarizados.

- `'ClassNames'`: Son los nombres de las clases que usamos para el entrenamiento. Tal como se puede ver en el Segmento de código 2.10 `'EXCELENTE CALIDAD'` y `'MALA CALIDAD'` son tipos de datos categóricos por lo tanto se configura este argumento como `categorical`.

Del mismo modo, se utilizó el método de validación cruzada k-fold (*K-fold cross-validation*) [27] para el parámetro k = 10 (es decir, 9 *folds* para el entrenamiento correspondiente a aproximadamente 222 imágenes y el resto para la prueba), cuyo proceso se describe a continuación:

1. Primero se define el número de *folds* y se obtienen las etiquetas de la base de datos `imds2` (tal como se mencionó al final de la Sección 2.1, `imds2` es la base de datos de las imágenes aisladas el fondo). De igual forma, hay que crear la partición de datos

para k-*fold cross-validation* a través del comando `cvpartition`, tal como se puede ver en el Segmento de código 2.12.

Segmento de código 2.12: Uso del comando cvpartition

```
% Se crea la partición para la validación cruzada de los …
    datos
cvp = cvpartition(etiq_imds2, 'KFold', K_Folds);
```

donde, `cvpartition` posibilita la creación de una partición aleatoria de nuestra base de datos.

- `etiq_imds2`: Son las etiquetas de la base de datos `imds2` que se consiguen digitando `imds2.Labels` (i.e. colocando .Labels como sufijo de `imds2`).
- `'KFold'`: Se refiere al número de particiones en que se dividirán los datos y en nuestro caso es 10.

2. Luego, se realiza un procedimiento iterativo en cada fold de prueba, donde se optimizarán las pérdidas o error de validación cruzada.

3. Dentro del procedimiento iterativo se obtienen los datos y las etiquetas de las imágenes del fold de entrenamiento (i.e. `imds2_entrenamiento_fold.Files` e `imds2_entrenamiento_fold.Labels` respectivamente).

   Además, se aplica el objeto `bagOfFeatures` sobre las imágenes de entrenamiento `imds2_entrenamiento_fold` para construir la Bolsa de Palabras Visuales (BoVW) `Bolsa1` (ver Segmento de código 2.9).

4. Posteriormente, se codifican las imágenes de entrenamiento con la bolsa de palabras visuales determinada a través del comando `encode`.

   A continuación, en el Segmento de código 2.13 se indica el uso y los argumentos del comando `encode`:

Segmento de código 2.13: Uso del comando encode

```
    % Se codifican las imágenes del datastore para el …
       entrenamiento
    Mat_caract_entrenamiento = encode(Bolsa1, …
       imds2_entrenamiento_fold);
```

- En MATLAB, `encode` permite crear un vector de características el cual representa un histograma de ocurrencias de las palabras visuales.
- `Bolsa1`: Es el objeto Bolsa de Palabras Visuales (ver Segmento de código 2.9).

- `imds2_entrenamiento_fold`: Son las imágenes de entrenamiento especificadas como un objeto ImageDatastore.
- `Mat_caract_entrenamiento`: Este argumento de salida indica el histograma de las palabras visuales.

5. Seguidamente, se entrena el clasificador SVM lineal con el vector de características de entrenamiento y las etiquetas de las imágenes de entrenamiento, tal como se mostró en el Segmento de código 2.10 y se obtiene un modelo de clasificador SVM.

6. Después, se extraen los datos y las etiquetas de las imágenes del fold de prueba (i.e. `imds2_prueba_fold.Files` e `imds2_prueba_fold.Labels` respectivamente) para que de igual forma se codifiquen a las imágenes de prueba con la bolsa de palabras visuales dando como resultado el vector de características de prueba `Mat_caract_prueba`.

7. Tras la obtención del modelo SVM y del vector de características de prueba, se realiza la predicción para generar las predicciones fold `Predicciones_fold` y los puntajes fold `Puntajes_fold`.

8. Por último, se guardan tanto las predicciones como los puntajes fold en las variables: predicciones de validación y puntajes de validación i.e. `Predictores_validacion` y `Puntajes_validacion` respectivamente.

Por otra parte, en esta sección también se usó el Área Bajo la Curva ROC (AUC) para medir el rendimiento del clasificador a través de las curvas Características Operativas del Receptor (*Receiver Operating Characteristic*, ROC) (ver 1.3.3). Para calcular este rendimiento se utiliza como argumentos las predicciones de validación, los puntajes de validación y las etiquetas de la base de datos.

3. RESULTADOS Y DISCUSIÓN (APLICACIÓN METODOLÓGICA)

Dentro de este capítulo se presentan los resultados obtenidos de la remoción del fondo de las imágenes con k-means. Igualmente, se indican los resultados de la utilización del clasificador SVM lineal y gaussiano en la base de datos de las imágenes de granos de cacao frescos (con pulpa), empleando *k-fold cross validation* y métricas tales como, la exactitud, la precisión, entre otras para evaluar el rendimiento del clasificador.

3.1. RESULTADOS OBTENIDOS EN LA SEGMENTACIÓN Y REMOCIÓN DEL FONDO CON K-MEANS

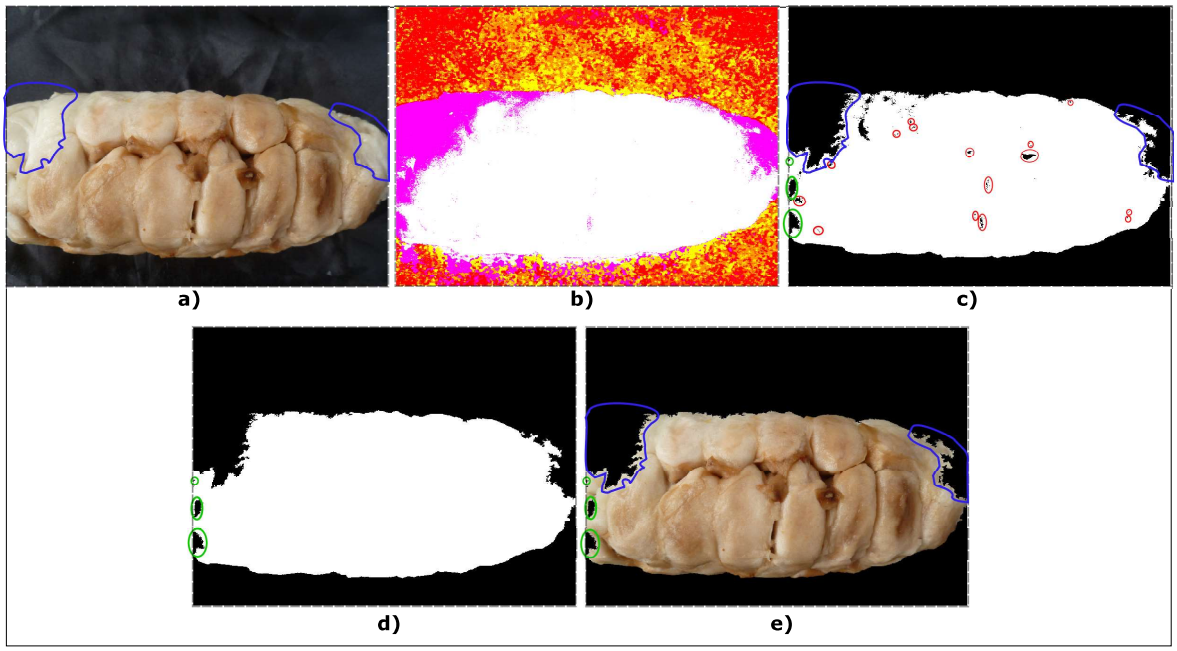


Figura 3.1: Ejemplo 1 de las etapas de remoción del fondo de una imagen de EXCELENTE CALIDAD, donde la segmentación k-means no alcanzó buenos resultados.

La Figura 3.1 muestra el antes (ver Figura 3.1.a) y el después (ver Figura 3.1.e) de una imagen de granos de cacao frescos de EXCELENTE CALIDAD aislada el fondo, donde la segmentación con el algoritmo k-means produce una confusión entre una porción de la fruta y el fondo. Por ejemplo, en la Figura 3.1.c que es el resultado de ejecutarse la segmentación y los pasos para determinar el clúster correcto que contiene a la fruta, vemos que la confusión antes mencionada es lo que se encuentra encerrado en color azul.

La Figura 3.1.b representa a la imagen convertida al espacio CIE 1976 L*a*b* y de alguna forma nos da una idea de que un fragmento del área dentro de la fruta se mezclará con el fondo en la segmentación.

Con respecto a la Figura 3.1.d, es notable que los huecos seleccionados en color rojo de

la Figura 3.1.c desaparecen. Sin embargo, se puede observar que las áreas encerradas en color verde que se asemejan también a huecos, se mantienen.

En efecto, las áreas encerradas en verde no son consideradas como huecos por el comando `imfill` debido a que son píxeles en negro que están dentro de un área blanca pero parten desde un borde de la imagen, tal como se mencionó en las secciones anteriores.

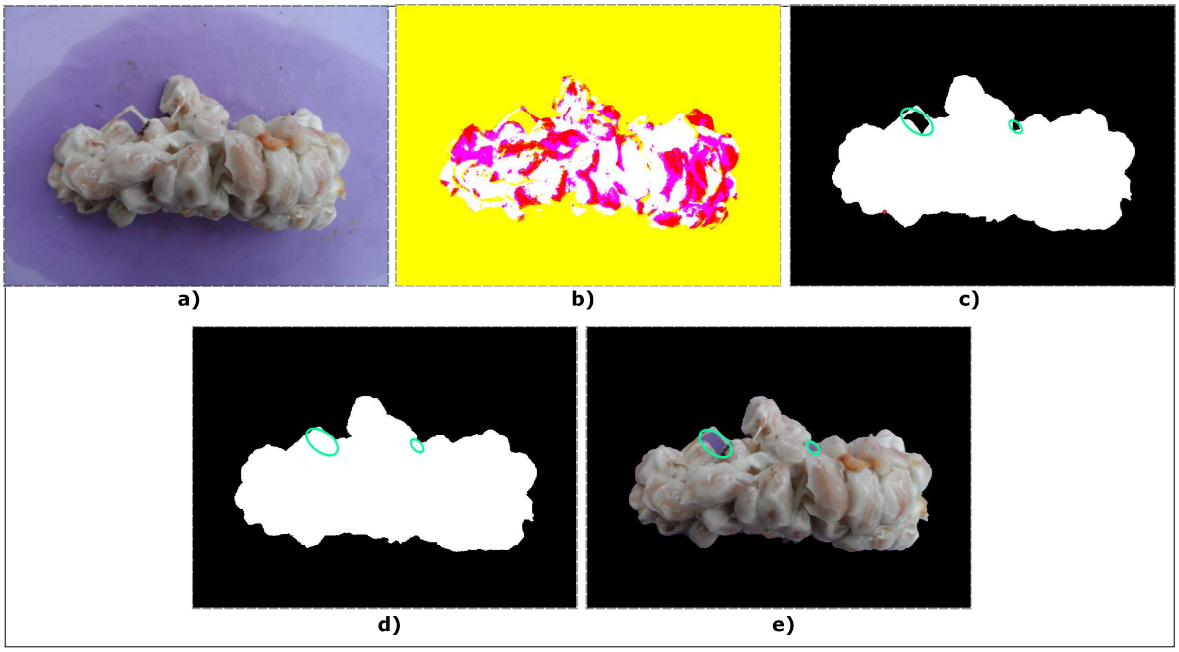


Figura 3.2: Ejemplo 2 de las etapas de remoción del fondo de una imagen de EXCELENTE CALIDAD, donde el llenado de huecos no es muy conveniente.

Ahora bien, si observamos la Figura 3.2 evidenciamos que el resultado (ver Figura 3.2.e) del proceso de quitar el fondo de otro ejemplo de una imagen de granos de cacao frescos de EXCELENTE CALIDAD (ver Figura 3.2.a), tiene el problema de que hay parte del fondo dentro de la fruta que no se ha eliminado. Estas áreas que pertenecen al fondo y que se han mantenido hasta la imagen final (ver Figura 3.2.e), se señalan en color turquesa.

Según lo que se puede apreciar desde la Figura 3.2.a hasta la Figura 3.2.c, la segmentación es muy buena. Inclusive, se observa que a pesar de que el fondo en la Figura 3.2.a no es uniforme debido a que parte de la superficie está mojada, el algoritmo k-means ha conseguido discriminar bien este fondo.

Por consiguiente, en la Figura 3.2.d que es el paso de llenado de huecos, es donde se da el inconveniente porque se se llenaron como huecos las áreas encerradas en turquesa pertenecientes al fondo.

La decisión de aplicar `imfill` para llenar huecos es acertada, pero en casos específicos como el de la Figura 3.2 no resulta muy beneficioso.

Por otra parte, en la Figura 3.3, se tiene un ejemplo de una imagen de granos de cacao frescos de MALA CALIDAD que ha pasado por las etapas de remoción del fondo y se observa que la segmentación ocasiona que se pierda parte del fruto.

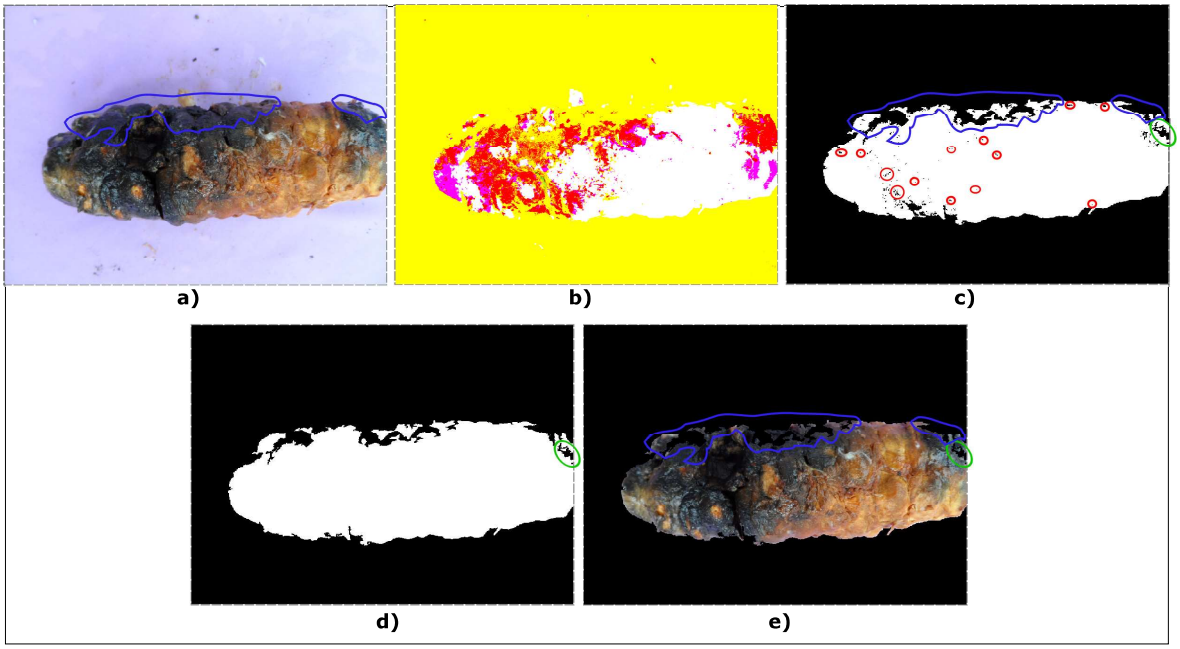


Figura 3.3: Ejemplo 3 de las etapas de remoción del fondo de una imagen de MALA CALIDAD, donde la segmentación k-means no obtuvo buenos resultados.

De manera análoga a lo acontecido en la Figura 3.1, se puede evidenciar en la Figura 3.3.c el resultado de la segmentación donde se encierra en azul el área que se confunde con el fondo. De igual forma, en la Figura 3.3.d se aprecia que el área encerrada en verde no corresponde a un hueco.

Se observa en la Figura 3.1 y en la Figura 3.3 que los granos de cacao frescos terminan en los bordes de la imagen provocando que no se puedan llenar los huecos presentes en aquellos extremos.

Seguidamente, en la Figura 3.4, se presenta otro caso del proceso de remoción del fondo de una imagen de granos de cacao frescos de MALA CALIDAD. La Figura 3.4.a tiene la particularidad de que en ella hay un grano de cacao (encerrado en color azul) que está apartado del conjunto más grande de estos granos y según lo que se puede observar en la imagen resultante del proceso (ver Figura 3.4.a), al remover el fondo, también se removió el grano apartado mencionado.

De acuerdo a la Figura 3.4.c, el grano fresco de cacao apartado que seguía presente hasta la Figura 3.4.b, desaparece. Esto se produce porque después de la segmentación con k-means, se extrae solo el objeto más grande en píxeles blancos mediante el comando `bwareafilt`.

Del mismo modo que en el ejemplo de la Figura 3.2, en la Figura 3.2.c se indica encerrado en turquesa un hueco que representa al fondo. Este hueco dentro de la fruta fue discriminado correctamente por k-means, sin embargo, como en el caso presentado de la Figura 3.2, no se puede elegir individualmente que hueco se llena y que hueco no.

En resumen, el uso del algoritmo k-means segmentado las imágenes para el propósito de remover el fondo de ellas, tuvo excelentes resultados exceptuando casos específicos como

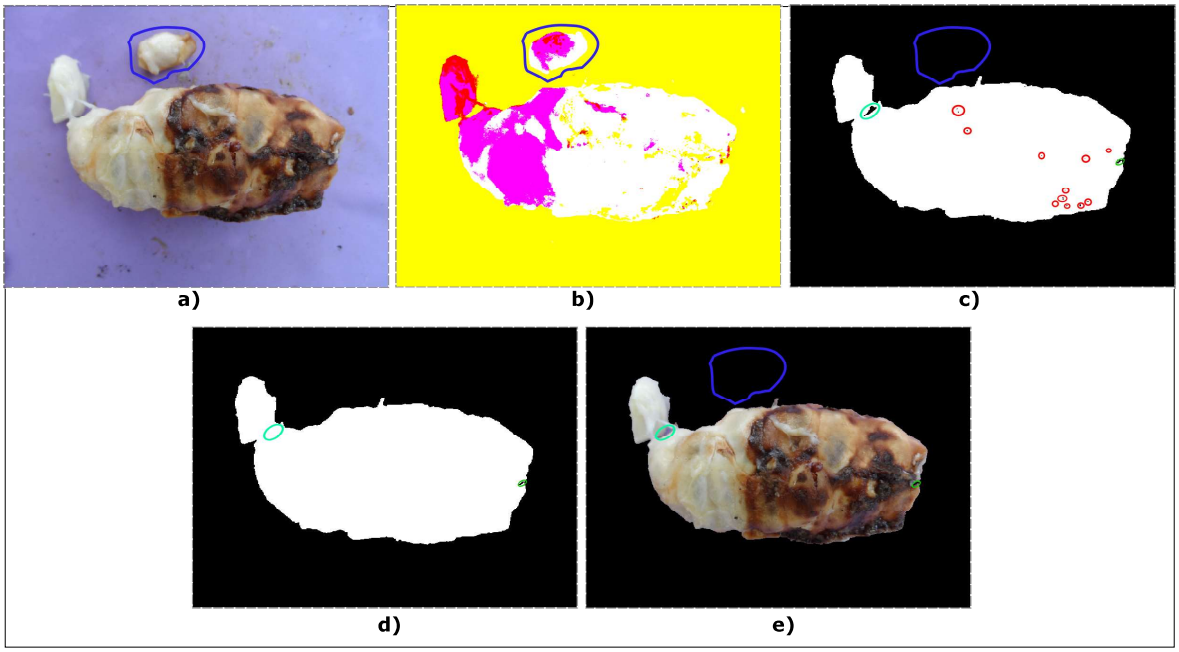


Figura 3.4: Ejemplo 4 de las etapas de remoción del fondo de una imagen de MALA CALIDAD, donde se pierde un grano de cacao por la extracción del objeto más grande en blanco.

en la Figura 3.1 y en la Figura 3.3.

Ciertamente, se requirieron pasos adicionales para determinar donde estaba la fruta y un paso de llenado de huecos que en casos muy extremos no es favorable, tal como se observó en la Figura 3.2 y en la Figura 3.4.

3.2. RESULTADOS OBTENIDOS EN LA UTILIZACIÓN DEL CLASIFICADOR SVM

La Figura 3.5 muestra un ajuste de baja dimensión usando la Incrustación de Vecinos Estocásticos Distribuidos en t (*t-distributed Stochastic Neighbor Embedding*, t-SNE). Para empezar, cuando tenemos datos en un espacio de características de alta dimensión, estos datos no se pueden visualizar de modo que se necesita de alguna herramienta para reducir su dimensionalidad y así poder mapear las características de alta dimensión en características de baja dimensión.

En el presente trabajo, el espacio de características es de 512 dimensiones. Entonces, con el propósito de visualizar este espacio de características, se tiene la técnica de visualización t-SNE [30] que incorpora los vectores de características de alta dimensión a dos dimensiones, los cuales sí se pueden visualizar fácilmente en un plano 2D.

Básicamente, este algoritmo calcula una incrustación de baja dimensión intentando preservar las similitudes de las distribuciones de las características de alta y baja dimensión [30]. En efecto, la técnica t-SNE está relacionada con la visualización de características de alta dimensión y no es dependiente del clasificador, por lo que en este trabajo la Figura 3.5 es para ambos clasificadores (i.e. el clasificador SVM lineal y gaussiano).

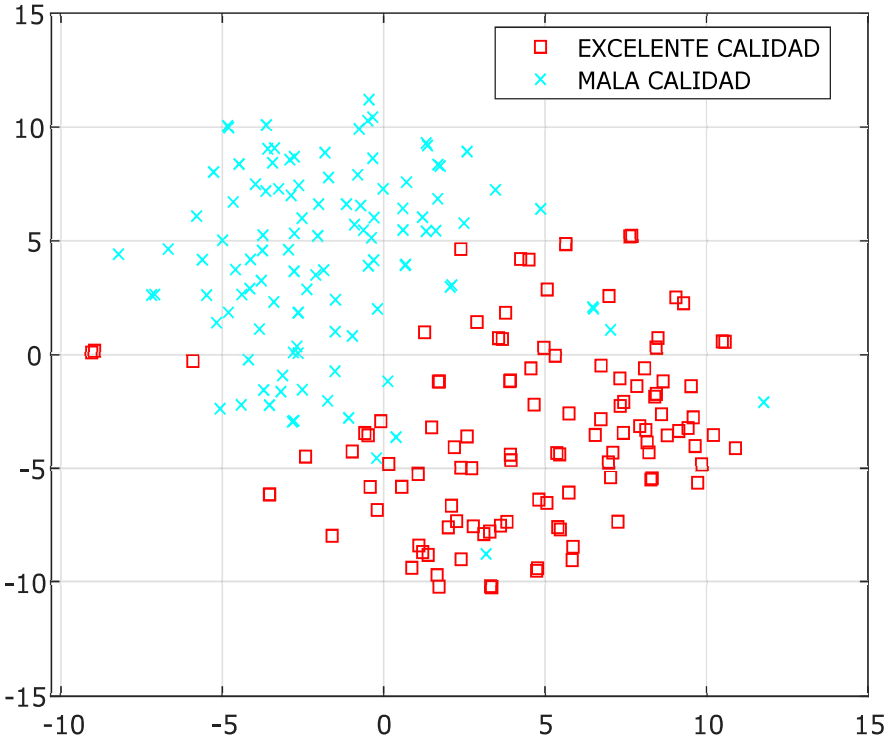


Figura 3.5: Plano de visualización t-SNE para las dos clases de calidad de granos de cacao frescos.

De acuerdo a los resultados de la Figura 3.5, se observa que aproximadamente las muestras se pueden separar linealmente, lo que justifica nuestra elección del núcleo lineal (*linear kernel*) en el clasificador SVM.

Por otra parte, a pesar de que se realizaron pruebas en el clasificador SVM con la función de núcleo lineal (*Linear kernel function*) y con la función de núcleo gaussiano (*Gaussian kernel function*), a través de la función de núcleo lineal se alcanzó un mejor rendimiento y fue más rápida que la gaussiana [23], [41].

Con la intención de evidenciar las pruebas expuestas del párrafo anterior, se indican los siguientes resultados:

La Figura 3.6 muestra los resultados de la optimización Bayesiana de hiperparámetros empleando la función de núcleo lineal. Específicamente, la Figura 3.6.a indica el resultado de la primera iteración (i.e. fold = 1), la Figura 3.6.b se refiere a la quinta iteración (i.e. fold = 5) y la Figura 3.6.c es de la décima iteración (i.e. fold = 10).

La Figura 3.6.a señala que el clasificador SVM lineal obtuvo aproximadamente un 2.24 % de pérdidas por validación cruzada para fold = 1, en la Figura 3.6.b se alcanza aproximadamente un 2.25 % de pérdidas con fold = 5 y finalmente en la Figura 3.6.c con fold = 10 las pérdidas llegan a un valor aproximado de 2.24 %. Según estos resultados, para fold = 1 y fold = 10 se tienen menos pérdidas en comparación con fold = 5 aunque la diferencia entre ellos no sea muy notoria.

Cabe señalar que, en cada iteración o *fold* se están evaluando 30 funciones objetivo para minimizar las pérdidas de validación cruzada. En efecto, se realizan evaluaciones hasta encon-

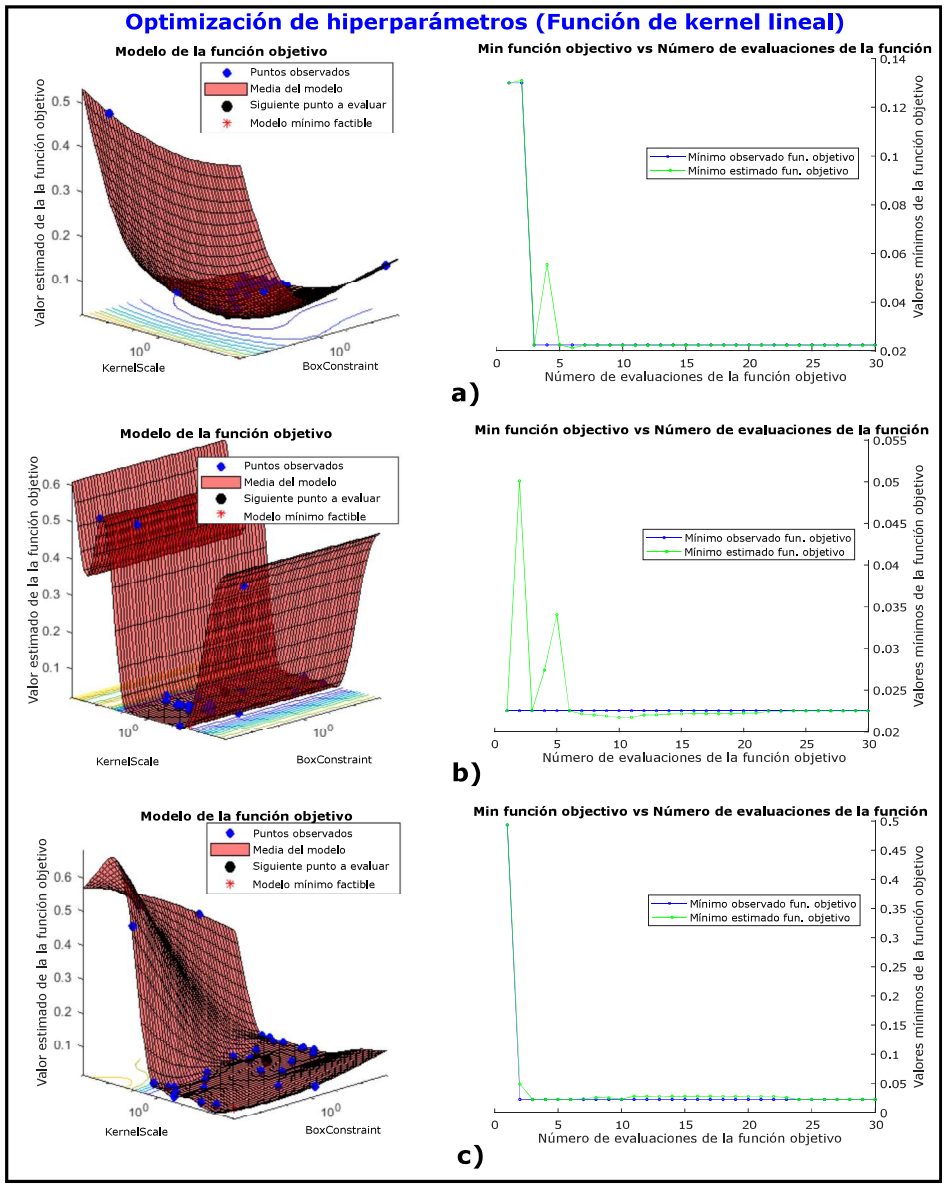


Figura 3.6: Resultados de la optimización de hiperparámetros para la función de núcleo o kernel lineal durante la primera iteración a), quinta iteración b) y décima iteración c).

trar el mejor valor entre los parámetros *KernelScale* y *BoxConstraint* (ver Figuras 3.6, 3.7). Donde *KernelScale* es el tamaño del kernel (se refiere a qué tan dispersos están los datos) mientras que *BoxConstraint* es la constante de penalización (llamada también como regularización) que evita el sobre ajuste (*overfiting*).

Continuando con la Figura 3.7, se pueden observar los resultados de la optimización de hiperparámetros para el clasificador SVM con la función de núcleo gaussiano, donde la Figura 3.7.a representa a la primera iteración, la Figura 3.7.b es la quinta iteración y la Figura 3.7.c corresponde a la décima iteración.

La Figura 3.7.a logra un 3.13 % de pérdidas por validación cruzada, en la Figura 3.7.b tenemos un 3.15 % de pérdidas y en la Figura 3.7.c se consigue un 1.7 % de estas pérdidas. Por consiguiente, la menor pérdida obtenida de la validación cruzada entre la primera, quinta y

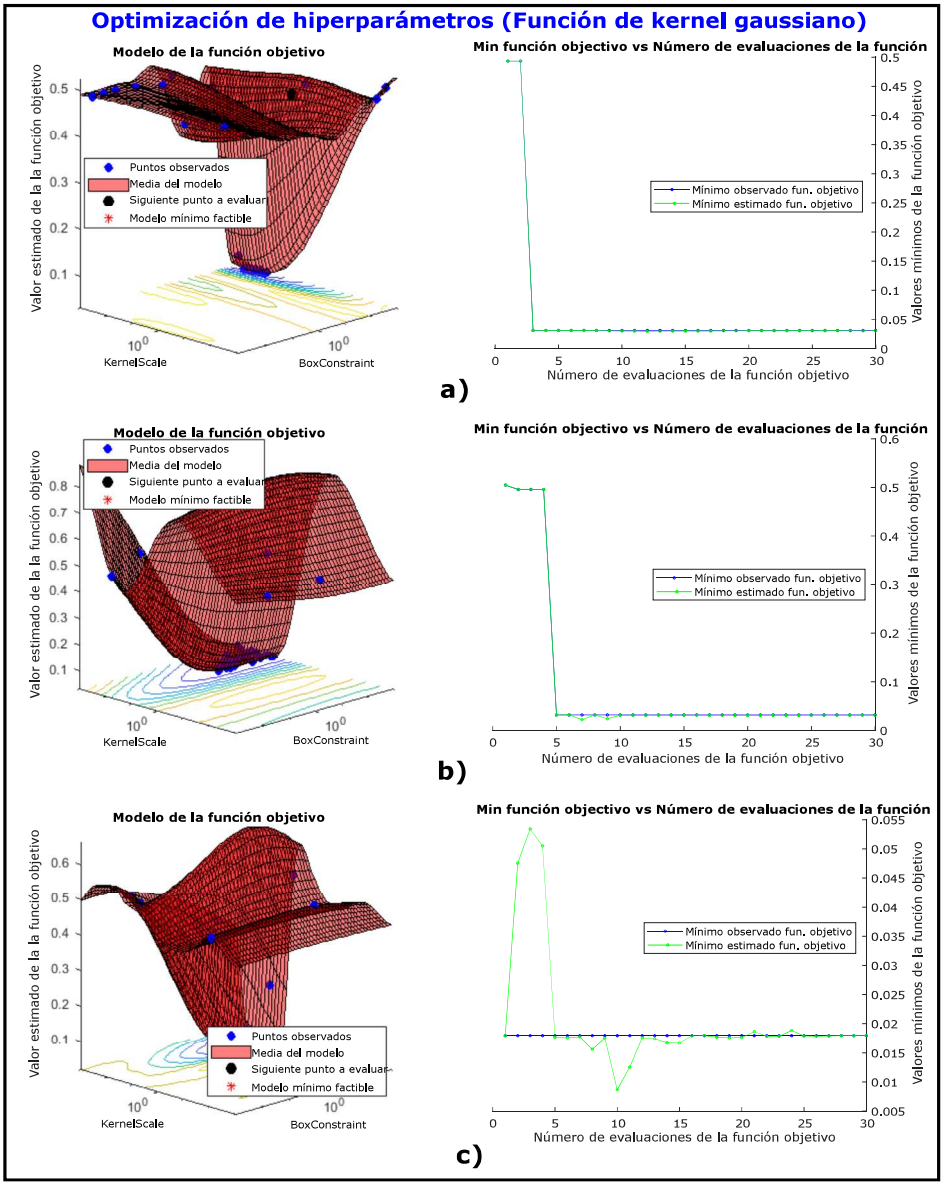


Figura 3.7: Resultados de la optimización de hiperparámetros para la función de núcleo o kernel gaussiano durante la primera iteración a), quinta iteración b) y décima iteración c).

décima iteración, corresponde a la décima iteración (i.e. fold = 10).

De hecho, los resultados de la Figura 3.6 y de la Figura 3.7, permiten evidenciar que al concluirse la iteración 10 del proceso de *k-fold cross validation* en ambas funciones, la menor perdida para la validación cruzada se encuentra en la función de núcleo gaussiano (ver Figura 3.7.c). Sin embargo, la diferencia de los porcentajes de pérdidas alcanzados con las dos funciones no es tan significativo, además conviene destacar que un clasificador SVM gaussiano es un proceso más complejo y demanda más tiempo con relación al clasificador SVM lineal [23].

Ahora bien, en la Figura 3.8 se pueden observar las curvas ROC de las funciones de núcleo lineal y gausiano del clasificador SVM utilizado en este trabajo. Como resultado, se aprecia gráficamente que el AUC que representa a la función de núcleo lineal es mayor que el AUC

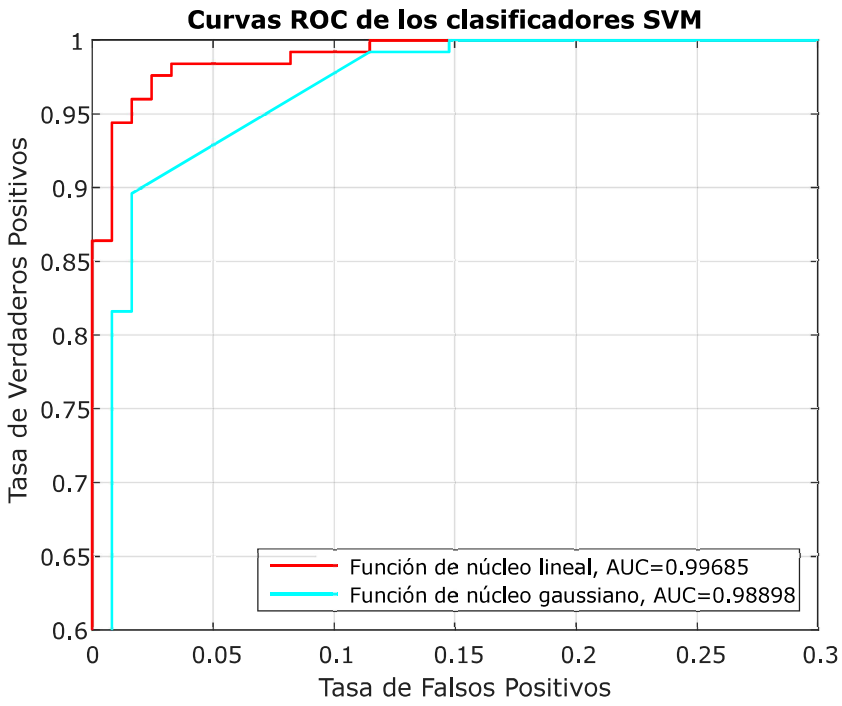


Figura 3.8: Curvas ROC para las funciones de núcleo lineal y gaussiano. En la validación cruzada se empleó 10 folds y la clase positiva es “EXCELENTE CALIDAD”.

de la función de núcleo gaussiano. Esto quiere decir que el rendimiento del clasificador SVM lineal es el mejor.

Concretamente, en la Figura 3.8 se puede ver que la curva ROC para la función de núcleo lineal muestra un Área Bajo la Curva ROC (AUC) que es de 99.68 %. Por ejemplo, para una Tasa de Falsos Positivos del 5 %, se logró aproximadamente el 98 % de la Tasa de Verdaderos Positivos.
En la Figura 3.9, se puede ver la matriz de confusión del clasificador SVM lineal utilizado. La matriz de confusión [42] es un método simple de evaluación que permite describir el rendimiento de un clasificador a través de sus parámetros. En esta matriz, las filas son las Clases verdaderas, es decir, nuestras observaciones originales.

Por otro lado, en las columnas se encuentran las Clases pronosticadas, por ejemplo, en nuestro caso sería la predicción del clasificador, es decir, qué observaciones se pronosticaron como excelente calidad y qué observaciones se pronosticaron como mala calidad.

Teniendo en cuenta eso, se utilizó una clasificación binaria, entonces la primera celda representa los valores de Verdaderos Positivos (*True Positives*, TP), que es la cantidad de positivos que se clasificaron correctamente como positivos.

En la cuarta celda se encuentran los Verdaderos Negativos (*True Negatives*, TN) que indican la cantidad de negativos que se clasificaron correctamente como negativos.

Seguidamente, las dos celdas restantes representan los Falsos Negativos (*False Negatives*, FN) y los Falsos Positivos (*False Positives*, FP), los cuales indican la cantidad de positivos o negativos que se clasificaron incorrectamente como negativos o positivos respectivamente.

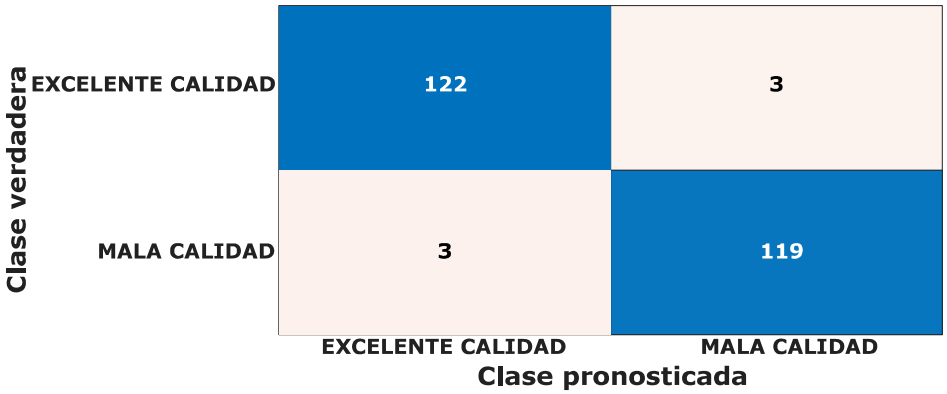


Figura 3.9: Matriz de confusión para la clasificación de granos de cacao frescos con el clasificador SVM lineal.

Con los valores de nuestra matriz de confusión del clasificador SVM lineal, se calculó las métricas de exactitud (*accuracy*), precisión (*precision*), sensibilidad (*recall*), especificidad (*specificity*) y puntuación-f1 (*f1-score*) [43] a través de las Ecuaciones 3.1, 3.2, 3.3, 3.4, 3.5:

$$\text{Exactitud} = \frac{TP + TN}{P + N} \tag{3.1}$$

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{3.2}$$

$$\text{Sensibilidad} = \frac{TP}{P} \tag{3.3}$$

$$\text{Especificidad} = \frac{TN}{N} \tag{3.4}$$

$$\text{Puntuación-f1} = \frac{2(\text{Precisión} \cdot \text{Sensibilidad})}{\text{Precisión} + \text{Sensibilidad}}, \tag{3.5}$$

donde, P es el número de muestras positivas y N es el número de muestras negativas.

Como resultado, se observa en la matriz de confusión que 122 imágenes de granos de cacao frescos de EXCELENTE CALIDAD se clasifican correctamente como EXCELENTE CALIDAD. Del mismo modo, 119 imágenes de granos de cacao frescos de MALA CALIDAD se clasifican correctamente como MALA CALIDAD.

Notar en la Tabla 3.1 que nuestra proporción de positivos reales que se predicen positivos (Sensibilidad) es de 97.6 %, nuestra proporción de negativos reales que se predicen como negativos (Especificidad) es de 97.54 % y nuestra media armónica entre la precisión y la sensibilidad (i.e., la puntación-f1) es del 97,6 %. Finalmente, la exactitud del clasificador SVM lineal es del 97.57 %, indicando un muy buen desempeño de este clasificador que utiliza aprendizaje supervisado.

Con respecto a trabajos anteriores, es difícil comparar nuestros resultados con los de ellos porque ninguno de los estudios previos trabaja con granos de cacao con pulpa, por lo que al proponer este manuscrito se pretende difundir esta forma de determinar la calidad de los

Tabla 3.1: Métricas obtenidas de la matriz de confusión del clasificador SVM lineal

Exactitud	97.57 %
Precisión	97.6 %
Sensibilidad	97.6 %
Especificidad	97.54 %
Puntuación-f1	97.6 %

granos de cacao.

Además, se puede examinar que en trabajos anteriores utilizaron bases de datos entre 35 y 175 imágenes. Mientras tanto en el presente trabajo, se empleó una base de datos más grande compuesta por 247 imágenes.

Para finalizar, un agricultor de cacao puede elegir el sistema que se ha planteado en este proyecto cuando quiera trabajar con un punto de operación específico de la curva AUC, donde se puede establecer una tasa baja de falsos positivos y consecuentemente determinar la tasa de verdaderos positivos. En definitiva, como se apreció en la Figura 3.8, este trabajo es una excelente opción debido a que es extremadamente confiable y automático.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

En el presente trabajo de titulación, se propuso un sistema de clasificación de granos de cacao frescos (con pulpa) que emplea el método *k-means clustering* y operaciones morfológicas para remover el fondo de las imágenes, luego se extrajo las características de estas imágenes con el modelo BoVW y finalmente para la clasificación, se utilizó un clasificador SVM con *k-fold cross-validation*.

Al comparar los clasificadores SVM para las funciones de núcleo gaussiano y lineal, mediante el gaussiano se obtuvo un *performance* ligeramente menor al lineal y nos quedamos con este último debido a que es más rápido ya que su kernel es simplemente una operación lineal [41].

Cuando se trabaja con los granos de cacao con pulpa, estos pueden mojar el fondo, lo cual puede ser un inconveniente para la segmentación. Sin embargo, se observó que el fondo cuando está mojado no influye en nuestra segmentación haciéndola robusta para este caso.

Ahora bien, si la fruta y el fondo tienen mucho brillo, la segmentación empieza a tener problemas de modo que para este caso la segmentación ya no es robusta.

Por otra parte, cuando la fruta no está capturada completamente en la imagen pueden existir problemas en los bordes. Por ejemplo, si hay píxeles negros que empiezan en los bordes y están contenidos dentro de la fruta (en píxeles blancos), el llenado de huecos (para esos píxeles negros) ya no se efectúa porque que en ese caso no se los considera como tal.

El sistema propuesto es robusto cuando las fotos del cacao tienen granos separados ya que finalmente se toma al objeto mayor, lo cual es bueno.

El modelo de Bolsa de Palabras Visuales (BoVW) usado, se basó en el descriptor SURF que es invariante a la rotación y traslación. Sin embargo, este descriptor no es invariante a las transformaciones de proyección y afines, lo que implica que la cámara debe estar en un lugar fijo, frente a la fruta.

Los resultados indican que se logró obtener un gran rendimiento del clasificador SVM lineal, donde el AUC fue de 99.68 % y la precisión de 97.57 %. Consecuentemente, esto demuestra que las técnicas de visión computacional son prometedoras en el campo de la clasificación de granos de cacao frescos (con pulpa).

Este trabajo sería útil para los productores de cacao en el proceso de despulpe de esta fruta, ya que lograrían una automatización de la producción y reducirían la intervención de las personas que realizan esta actividad. Al mismo tiempo, esto evitaría el margen de error introducido por los métodos tradicionales basados en los órganos de los sentidos de los agricultores, lo que a su vez permitiría obtener un método estándar de determinación de la calidad independiente del evaluador.

4.2. RECOMENDACIONES

Dado que los algoritmos utilizados en el enfoque de este proyecto no son computacionalmente complejos, los trabajos futuros podrían implementar estas ideas en una aplicación móvil o sistema embebido para evaluar la calidad de los granos de cacao frescos in situ.

Asimismo, considerar para la aplicación móvil que como en el presente trabajo se utilizó una cámara fija y en un dispositivo móvil la cámara no necesariamente va a tomar una foto de frente a la fruta, esto va a generar distorsiones en la imagen. Aquellas distorsiones serán del tipo proyectivas y afines, por lo que SURF no es robusto ante ellas. En ese caso, es recomendable que se examine otro descriptor, por ejemplo, el descriptor MSER [44].

Es importante mencionar que para ayudar a la segmentación con el problema suscitado por parte brillo, se puede ecualizar a las imágenes para que así mejoren su contraste.

Del mismo modo, se sugiere que en otros trabajos futuros se apliquen diferentes técnicas de aprendizaje de máquinas, por ejemplo, se puede emplear el aprendizaje profundo (*deep learning*) [45]. Ciertamente, *deep learning* es un *framework* donde el algoritmo aprende las características por sí mismo y esto lo hace diferente de nosotros ya que aquí se utilizó un extractor de características denominado *bag of features*.

Es recomendable que en la implementación futura de este trabajo, se tenga un prototipo con una cámara fotográfica aislada. Igualmente, en prototipos futuros es importante considerar las condiciones ambientales, para que se tenga una buena iluminación independientemente de que sea día o noche y con esto lograr que las imágenes se procesen correctamente.

Por otra parte, en la implementación práctica, se puede añadir la opción de subir los datos generados por el clasificador hacia un servidor y de igual forma establecer un método de alerta (mediante notificaciones) de los productos no deseables para el agricultor.

Se sugiere en otros trabajos futuros, realizar una comparativa entre clasificadores basados en SVM, redes neuronales y lógica difusa.

En definitiva, se pueden extender estas ideas propuestas en el presente trabajo de titulación para clasificar no únicamente la pulpa del cacao, sino también las mazorcas determinando si están infectadas o no, así como también los granos de cacao sin pulpa verificando su grado de fermentación.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] J. S. Henderson, R. A. Joyce, G. R. Hall, W. J. Hurst, and P. E. McGovern, "Chemical and archaeological evidence for the earliest cacao beverages," *Proceedings of the National Academy of Sciences*, vol. 104, no. 48, pp. 18 937–18 940, 2007.

[2] R. G. L. Solorzano, O. Fouet, A. Lemainque, S. Pavek, M. Boccara, X. Argout, F. Amores, B. Courtois, A. M. Risterucci, and C. Lanaud, "Insight into the wild origin, migration and domestication history of the fine flavour nacional theobroma cacao l. variety from ecuador," *PLoS One*, vol. 7, no. 11, p. e48438, 2012.

[3] J. Yi, L. Zhou, J. Bi, Q. Chen, X. Liu, and X. Wu, "Influence of pre-drying treatments on physicochemical and organoleptic properties of explosion puff dried jackfruit chips," *Journal of food science and technology*, vol. 53, no. 2, pp. 1120–1129, 2016.

[4] C. E. Hansen, M. del Olmo, and C. Burri, "Enzyme activities in cocoa beans during fermentation," *Journal of the Science of Food and Agriculture*, vol. 77, no. 2, pp. 273–281, 1998.

[5] M. O. K. D. Albarracin Macias Miguel Eduardo and G. C. J. andrés, *Diseño y simulación de una maquina clasificadora por visión artificial y despulpadora de cacao*. Quevedo: UTEQ, 2019. [Online]. Disponible en: http://biblioteca.uteq.edu.ec/cgi-bin/koha/opac-detail.pl?biblionumber=3704

[6] R. C. Gonzalez and E. Richard, "Woods, digital image processing," *ed: Prentice Hall Press, ISBN 0-201-18075*, vol. 8, 2002. [Online]. Disponible en: https://books.google.com.ec/books?id=YRRkQgAACAAJ

[7] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image segmentation using k-means clustering algorithm and subtractive clustering algorithm," *Procedia Computer Science*, vol. 54, pp. 764–771, 2015.

[8] X.-w. Lou, D.-c. Huang, L.-m. Fan, and A.-j. Xu, "An image classification algorithm based on bag of visual words and multi-kernel learning," *Journal of Multimedia*, vol. 9, no. 2, p. 269, 2014.

[9] V. S. Nalwa, *A guided tour of computer vision*. Addison-Wesley Longman Publishing Co., Inc., 1994.

[10] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[11] P. Soille, *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[12] M. M. Hasan and P. K. Mishra, "Improving morphology operation for 2d hole filling algorithm," *International Journal of Image Processing (IJIP)*, vol. 6, no. 1, pp. 635–646, 2012.

[13] H. Ng, S. Ong, K. Foong, P. Goh, and W. Nowinski, "Medical image segmentation using k-means clustering and improved watershed algorithm," in *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*. IEEE, 2006, pp. 61–65.

[14] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Computer Vision and Image Understanding*, vol. 150, pp. 109–125, 2016.

[15] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[16] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1. Prague, 2004, pp. 1–2.

[17] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2010, pp. 270–279.

[18] P. Panchal, S. Panchal, and S. Shah, "A comparison of sift and surf," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 323–327, 2013.

[19] C. M. Bishop, *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[20] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[21] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited„ 2016.

[22] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an em approach," in *Advances in neural information processing systems*, 1994, pp. 120–127.

[23] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[24] M. Hussain, S. K. Wajid, A. Elzaart, and M. Berbar, "A comparison of svm kernel functions for breast cancer detection," in *2011 Eighth International Conference Computer Graphics, Imaging and Visualization*. IEEE, 2011, pp. 145–150.

[25] G. Madzarov, D. Gjorgjevikj, and I. Chorbev, "A multi-class svm classifier utilizing binary decision tree," *Informatica*, vol. 33, no. 2, 2009.

[26] J. Huang, X. Shao, and H. Wechsler, "Face pose discrimination using support vector machines (svm)," in *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, vol. 1. IEEE, 1998, pp. 154–156.

[27] Y. Bengio and Y. Grandvalet, “No unbiased estimator of the variance of k-fold cross-validation,” *Journal of machine learning research*, vol. 5, no. Sep, pp. 1089–1105, 2004.

[28] A. P. Bradley, “The use of the area under the roc curve in the evaluation of machine learning algorithms,” *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[29] T. Fawcett, “An introduction to roc analysis,” *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[30] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[31] “3.6.10.5. tSNE to visualize digits — Scipy lecture notes.” [Online]. Disponible en: https://scipy-lectures.org/packages/scikit-learn/auto_examples/plot_tsne.html

[32] D. S. Tan, R. N. Leong, A. F. Laguna, C. Anne Ngo, A. Lao, D. M. Amalin, and D. Alvindia, “Autodidac: Automated tool for disease detection and assessment for cacao black pod rot,” *Crop Protection*, vol. 103, pp. 98–102, 01 2018.

[33] S. A. Veites-Campos, R. Ramírez-Betancour, and M. Gonzalez-Perez, “Identification of cocoa pods with image processing and artificial neural networks,” *International Journal of Advanced Engineering, Management and Science*, vol. 4, pp. 510–518, 07 2018.

[34] D. S. Tan, R. N. Leong, A. F. Laguna, C. A. Ngo, A. Lao, D. Amalin, and D. Alvindia, “A framework for measuring infection level on cacao pods,” in *2016 IEEE Region 10 Symposium (TENSYMP)*. IEEE, 2016, pp. 384–389.

[35] P. Parra, T. Negrete, J. Llaguno, and N. Vega, “Computer vision techniques applied in the estimation of the cocoa beans fermentation grade,” in *2018 IEEE ANDESCON*. IEEE, 2018, pp. 1–10.

[36] A. Lawi and Y. Adhitya, “Classifying physical morphology of cocoa beans digital images using multiclass ensemble least-squares support vector machine,” *Journal of Physics: Conference Series*, vol. 979, p. 012029, mar 2018. [Online]. Disponible en: https://doi.org/10.1088%2F1742-6596%2F979%2F1%2F012029

[37] A. Yro, C. E. N’zi, and K. Kpalma, “Cocoa beans fermentation degree assessment for quality control using machine vision and multiclass svm classifier,” *International Journal of Innovation and Applied Studies*, 2018.

[38] N. Leon-Roque, M. Abderrahim, L. Nuñez-Alejos, S. M. Arribas, and L. Condezo-Hoyos, “Prediction of fermentation index of cacao beans (theobroma cacao l.) based on color measurement and artificial neural networks,” *Talanta*, vol. 161, 08 2016.

[39] D. C. Obediencia, D. G. Brosas, and R. S. Villafuerte, “Cacao bean quality assessment procedure: A method for classification process,” *Journal of Computing and Innovation*, vol. 2, no. 2, pp. 60–73, 2018.

[40] P. J. Baldevbhai and R. Anand, “Color image segmentation for medical images using l* a* b* color space,” *IOSR Journal of Electronics and Communication Engineering*, vol. 1, no. 2, pp. 24–45, 2012.

[41] A. Ben-Hur and J. Weston, “A user’s guide to support vector machines,” in *Data mining techniques for the life sciences*. Springer, 2010, pp. 223–239.

[42] H. Lewis and M. Brown, “A generalized confusion matrix for assessing area estimates from remotely sensed data,” *International Journal of Remote Sensing*, vol. 22, no. 16, pp. 3223–3235, 2001.

[43] S. Visa, B. Ramsay, A. L. Ralescu, and E. Van Der Knaap, “Confusion matrix-based feature selection.” *MAICS*, vol. 710, pp. 120–127, 2011.

[44] R. Kimmel, C. Zhang, A. Bronstein, and M. Bronstein, “Are mser features really interesting?” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2316–2320, 2011.

[45] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

ANEXOS

ANEXO A. Código principal main_cocoa_classifier.m

ANEXO B. Función bovwfold_linear.m

ANEXO C. Función bovwfold_gaussian.m

ANEXO A: main_cocoa_classifier.m

```
% SISTEMA DE CLASIFICACIÓN DE GRANOS DE CACAO FRESCOS BASADOS EN VISIÓN COMPUTACIONAL.
%Jefferson Oña
%Programa principal
clc
close all
clear all

%% CREACIÓN DEL ALMACENAMIENTO DE DATOS (datastore) DE LAS IMÁGENES ORIGINALES DE CACAO.
imds1 = imageDatastore('C:\Users\Jeferson\Desktop\Tesis\Computer vision cacao\Ejemplos implementación MATLAB\Segmentación fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Originales\','FileExtensions','.png', ...
    'IncludeSubfolders',true,'LabelSource','foldernames');
% Se generan números aleatorios controlados.
rng('default')

%% CONFIGURACIONES PARA REMOVER EL FONDO DE LAS IMÁGENES.
for k = 1:length(imds1.Files)
    % Se lee la imagen especificada del data store para obtener información de la misma
    [img_ori1,info] = readimage(imds1, k);
    % Se redimensiona la imagen a un 25 %
    img_ori1 = imresize(img_ori1, 0.25);

    % SEGMENTACIÓN BASADA EN COLOR UTILIZANDO K-MEANS.
    cco = img_ori1;
    % Se convierte la imagen de color (RGB) al espacio CIE 1976 L*a*b*
    labsp_cco = rgb2lab(cco);
    ab_color = labsp_cco(:,:,2:3);
    % Se convierten los parámetros a* y b* a un tipo de datos sigle
    ab_sin = im2single(ab_color);

    % APLICACIÓN DE K-MEANS CLUSTERING
    % Se repite 2 veces el proceso de clustering k-means
    n_clusters = 2;
    % n_clusters: Indica el números de clústers
    % ab_sin: Corresponde a la imagen a segmentar
    % 'NumAttempts': Es el número de veces que se repite el proceso de
    % agrupación
    % 'NormalizeInput': Indica la normalización de los datos de entrada
```

```
37      % 'MaxIterations': Es el número máximo de iteraciones
38      % 'Threshold': Indica el umbra de precisión
39      etiqueta_pixel = imsegkmeans(ab_sin, n_clusters, ...
40          'NumAttempts', 2, 'NormalizeInput', false, ...
41          'MaxIterations', 100, 'Threshold', 5);
42
43      % DETERMINACIÓN DE LA MÁSCARA PARA QUITAR EL FONDO
44      % Se asigna a mascara1 el clúster = 1, que devuelve el comando …
           imsegkmeans
45      mascara1 = etiqueta_pixel == 1;
46      cluster1 = cco .* uint8(mascara1);
47      % Se asigna a mascara21 el clúster = 2, que devuelve el comando …
           imsegkmeans
48      mascara21 = etiqueta_pixel == 2;
49      cluster2 = cco .* uint8(mascara21);
50      % Se extrae el objeto más grande de la imagen segmentada para …
           ambos clusters
51      a1 = bwareafilt(mascara1,1);
52      a2 = bwareafilt(mascara21,1);
53      % Se obtiene las dimensiones del rectángulo que contiene al …
           objeto extraído
54      Bmask1 = regionprops(a1,'BoundingBox');
55      % Se determina el área del rectángulo que contiene al objeto
56      Amask1 = Bmask1.BoundingBox(3)*Bmask1.BoundingBox(4);
57      Bmask21 = regionprops(a2,'BoundingBox');
58      Amask21 = Bmask21.BoundingBox(3)*Bmask21.BoundingBox(4);
59
60      % ESTABLECIMIENTO DE LAS CONDICIONES PARA DETERMINAR LA MÁSCARA
61      % Se compara las áreas
62      if (Amask1<Amask21)
63          Bin_W2=a1;
64      else
65          Bin_W2=a2;
66      end
67      % Se llena los huecos de la imagen binaria
68      mascara = imfill(Bin_W2,'holes');
69      % Se multiplica la imagen con la máscara obtenida, para tener la …
           imagen removida el fondo
70      cluster_rem = cco .* uint8(mascara);
71      c = cluster_rem;
72
73      % CONFIGURACIONES PARA ALMACENAR LA IMAGEN REMOVIDA EL FONDO EN SU
74      % RESPECTIVA CARPETA
75      if info.Label=='EXCELENTE CALIDAD'
76          % Se remueve la extensión '.png' de las imágenes
77          nombre = erase(info.Filename,'.png');
78          % Se especifica la dirección de donde se obtendrá la imagen
```

```
79          nombre = ...
                erase(nombre,'C:\Users\Jeferson\Desktop\Tesis\Computer ...
                vision cacao\Ejemplos implementación MATLAB\Segmentación ...
                fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Originales\EXCELENTE ...
                CALIDAD\');
80
81          % Se agrega la palabra seg y la extensión '.png' al nombre ...
                de la imagen
82          ruta = strcat(nombre,'seg','.png');
83          % Se especifica la dirección a dónde va la imagen
84          ruta = strcat('C:\Users\Jeferson\Desktop\Tesis\Computer ...
                vision cacao\Ejemplos implementación MATLAB\Segmentación ...
                fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Segmentada ...
                kmeans\EXCELENTE CALIDAD/',ruta);
85
86          % Se escribe la imagen en la dirección indicada
87          imwrite(c,ruta)
88      end
89      if info.Label=='MALA CALIDAD'
90          nombre = erase(info.Filename,'.png');
91          nombre = ...
                erase(nombre,'C:\Users\Jeferson\Desktop\Tesis\Computer ...
                vision cacao\Ejemplos implementación MATLAB\Segmentación ...
                fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Originales\MALA ...
                CALIDAD\');
92
93          ruta = strcat(nombre,'seg','.png');
94          ruta = strcat('C:\Users\Jeferson\Desktop\Tesis\Computer ...
                vision cacao\Ejemplos implementación MATLAB\Segmentación ...
                fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Segmentada ...
                kmeans\MALA CALIDAD/',ruta);
95
96          imwrite(c,ruta)
97      end
98  end
99
100 %% CREACIÓN DEL DATASTORE DE LAS IMÁGENES REMOVIDAS EL FONDO
101 imds2 = imageDatastore('C:\Users\Jeferson\Desktop\Tesis\Computer ...
        vision cacao\Ejemplos implementación MATLAB\Segmentación fuzzy ...
        c-means\FRFCM\FRFCM PAR MEJOR\Segmentada ...
        kmeans\','FileExtensions','.png', ...
102      'IncludeSubfolders',true,'LabelSource','foldernames');
103
104 %% SE LLAMA A LAS FUNCIONES ELABORADAS PARA APLICAR BAG OF VISUAL ...
        WORDS, K-FOLD CROSS VALIDATION Y EL CLASIFICADOR SVM
105 % Número de particiones en las que se dividen los datos
106 in_Kfold1 = 10;
```

```
% Función para el Kernel Lineal.
% Devuelve: respuesta1, Puntajes_validacion1, …
   Predictores_validacion1, Mat_caract_entrenam1 e imds2_etiq_entrenam
[respuesta1,Puntajes_validacion1,Predictores_validacion1, ...
    Mat_caract_entrenam1,imds2_etiq_entrenam] = …
        bovwfold_linear(imds2,in_Kfold1);

in_Kfold2=10;
% Función para el Kernel Gausiano.
% Devuelve: respuesta2 y Puntajes_validacion2
[respuesta2,Puntajes_validacion2] = …
   bovwfold_gaussian(imds2,in_Kfold2);

%% Gráfica de la matriz de confusión para la función kernel lineal
% Se calcula la matriz de confusión a partir de las etiquetas del …
   datastore
% imds2 y las etiquetas predichas
Mat_Conf = confusionmat(double(respuesta1),Predictores_validacion1);
% Se muestra la matriz de confusión
confusionchart(Mat_Conf);

% Cálculo de la exactitud del clasificador SVM
% Se suman los elementos de la diagonal de la matriz de confusión
Sum_tp_tn = sum(diag(Mat_Conf));
% Se suman todos los elementos de la matriz de confusión
Sum_tot = sum(Mat_Conf,'all');
% Fórmula para calcular la exactitud del clasificador
Accuracy_SVM =Sum_tp_tn/Sum_tot;

%% Gráficas ROC, AUC para las funciones Lineal y Gaussiana
% Función que devuelve los valores de las coordenadas X y Y de la …
   curva ROC
[X1,Y1,T1,AUC1] = …
   perfcurve(respuesta1,Puntajes_validacion1(:,1),'EXCELENTE …
   CALIDAD');
[X2,Y2,T2,AUC2] = …
   perfcurve(respuesta2,Puntajes_validacion2(:,1),'EXCELENTE …
   CALIDAD');
figure
% Se grafica la curva ROC para la función lineal
plot(X1,Y1,'r','LineWidth',1.2);
grid on
hold on
% Se grafica la curva ROC para la función gaussiana
plot(X2,Y2,'c','LineWidth',1.2);
% Se configuran los límites para los ejes X y Y
xlim([0,0.3]); ylim([0.6,1]);
```

```
145 legend(['Función de Kernel lineal, AUC=',num2str(AUC1)],['Función de …
      Kernel gaussiano, AUC=',num2str(AUC2)],'Location','Southeast')
146 xlabel('Tasa de Falsos Positivos')
147 ylabel('Tasa de Verdaderos Positivos')
148 title('Curvas ROC para las funciones de Kernel lineal y gaussiano')
149
150 %% Gráfica ROC, AUC  para la función lineal
151 figure
152 % Se grafica la curva ROC
153 plot(X1,Y1,'r','LineWidth',1.2);
154 grid on
155 % Se configuran los límites para los ejes X y Y
156 xlim([0,0.3]); ylim([0.6,1]);
157 legend(['AUC=',num2str(AUC1)],'Location','Southeast')
158 xlabel('Tasa de Falsos Positivos')
159 ylabel('Tasa de Verdaderos Positivos')
160 title('Curva ROC para la función de Kernel lineal')
161
162 %% Gráfica t-SNE
163 % Se obtiene una matriz de dos dimensiones del argumento que es de …
      alta
164 % dimensión
165 Y5 = tsne(Mat_caract_entrenam1);
166 figure
167 % Se grafica el diagrama de dispersión con los valores de la matriz …
      de dos dimensiones
168 gscatter(Y5(:,1),Y5(:,2),imds2_etiq_entrenam,'rc','sx')
169 grid on
170 title('Gráfica t-SNE')
```

ANEXO B: Función bovwfold_linear.m

```
function [etiq_imds2,Puntajes_validacion,Predictores_validacion, ...
    Mat_caract_entrenamiento,etiq_imds2_entrenamiento] = …
        bovwfold_linear(imds2,in_Kfold)
% Función clasificación-validación desarrollada para el Kernel lineal
% Donde:
% etiq_imds2, son las etiquetas del datastore imds2.
% Puntajes_validacion, son los puntajes de validación.
% Predictores_validacion, son los predictores de validación.
% Mat_caract_entrenamiento, es la matriz de características de la …
   parte de entrenamiento.
% etiq_imds2_entrenamiento, son las etiquetas de la parte de …
   entrenamiento
% del datastore imds2.
% imds2,in_Kfold, son los parámetros de entrada para la función.

imds2 = imageDatastore('C:\Users\Jeferson\Desktop\Tesis\Computer …
    vision cacao\Ejemplos implementación MATLAB\Segmentación fuzzy …
    c-means\FRFCM\FRFCM PAR MEJOR\Segmentada …
    kmeans\','FileExtensions','.png', ...
     'IncludeSubfolders',true,'LabelSource','foldernames');

% DATOS PARA K-FOLD CROSS-VALIDATION
% Se lee el valor correspondiente a la entrada in_Kfold del programa
% principal
K_Folds = in_Kfold;
% Se asigna el campo .Labels de la estructura imds2 a etiq_imds2
etiq_imds2 = imds2.Labels;
% Se crea la partición para la validación cruzada de los datos
cvp = cvpartition(etiq_imds2, 'KFold', K_Folds);
Predictores_validacion = NaN(size(etiq_imds2));
num_observaciones = size(etiq_imds2, 1);
num_clases = 2;
Puntajes_validacion = NaN(num_observaciones, num_clases);

% K-FOLD CROSS-VALIDATION
for fold = 1:K_Folds
    % OBTENCIÓN DE LAS ETIQUETAS DEL ENTRENAMIENTO
    idx_fold = cvp.training(fold);
    imds2_entrenamiento_fold = …
        imageDatastore('C:\Users\Jeferson\Desktop\Tesis\Computer …
        vision cacao\Ejemplos implementación MATLAB\Segmentación …
        fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Segmentada …
        kmeans\','FileExtensions','.png', ...
```

```
34          'IncludeSubfolders',true,'LabelSource','foldernames');
35      etiquetas_entrenamiento_fold = imds2_entrenamiento_fold.Labels;
36      imds2_entrenamiento_fold.Files = …
            {imds2_entrenamiento_fold.Files{cvp.training(fold)}}';
37      imds2_entrenamiento_fold.Labels = …
            etiquetas_entrenamiento_fold(cvp.training(fold));
38      etiq_imds2_entrenamiento = imds2_entrenamiento_fold.Labels;
39
40       % APLICACIÓN DEL OBJETO BOLSA DE PALABRAS VISUALES (BoVW)
41      % Se construye la bolsa de palabras visuales para la extracción de
42      % características y personalizando sus propiedades
43      Bolsa1 = bagOfFeatures (imds2_entrenamiento_fold, ...
44          'VocabularySize', 512, 'StrongestFeatures', 0.8, ...
45          'PointSelection', 'Detector');
46      % Se codifican las imágenes del datastore para el entrenamiento
47      Mat_caract_entrenamiento = encode(Bolsa1, …
            imds2_entrenamiento_fold);
48
49      % CLASIFICADOR SVM LINEAL
50      imds2_entrenamiento = imds2_entrenamiento_fold.Labels;
51       clasificacion_SVM = fitcsvm(...
52       Mat_caract_entrenamiento, ...
53       imds2_entrenamiento, ...
54       'KernelFunction', 'linear', ...                              % Se …
             especifica la función kernel como lineal
55       'OptimizeHyperparameters', 'auto', ...
56       'Standardize', true, ...
57       'ClassNames', categorical({'EXCELENTE CALIDAD'; 'MALA …
             CALIDAD'}));
58  [Puntajes_SVM_Clasificador,Parametros_puntajes] = …
        fitSVMPosterior(clasificacion_SVM);
59
60      % Se crea la estructura de resultados a través de la función …
            'predict'
61      svmPredictFcn = @(x) predict(Puntajes_SVM_Clasificador, x);
62
63      % OBTENCIÓN DE LAS ETIQUETAS PARA LA PRUEBA
64      imds2_prueba_fold = …
            imageDatastore('C:\Users\Jeferson\Desktop\Tesis\Computer …
            vision cacao\Ejemplos implementación MATLAB\Segmentación …
            fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Segmentada …
            kmeans\','FileExtensions','.png', ...
65          'IncludeSubfolders',true,'LabelSource','foldernames');
66      etiquetas_prueba_fold = imds2_prueba_fold.Labels;
67      imds2_prueba_fold.Files = …
            {imds2_prueba_fold.Files{cvp.test(fold)}}';
68      imds2_prueba_fold.Labels = etiquetas_prueba_fold(cvp.test(fold));
```

```
69      % Se codifican las imágenes del datastore para la prueba
70      Mat_caract_prueba = encode(Bolsa1,imds2_prueba_fold);
71
72      % Se calculan las validaciones de predicción
73      [Predicciones_fold, Puntajes_fold] = …
            svmPredictFcn(Mat_caract_prueba);
74      % Se guardan las predicciones
75      Predictores_validacion(cvp.test(fold), :) = Predicciones_fold;
76      Puntajes_validacion(cvp.test(fold), :) = Puntajes_fold;
77  end
```

ANEXO c: Función bovwfold_gaussian.m

```
function [etiq_imds2g,Puntajes_validaciong] = bovwfold_gaussian(imds2,in_Kfold)
% Función clasificación-validación desarrollada para el Kernel gaussiano
% Donde:
% etiq_imds2g, son las etiquetas del datastore imds2.
% Puntajes_validaciong, son los puntajes de validación.

imds2 = imageDatastore('C:\Users\Jeferson\Desktop\Tesis\Computer vision cacao\Ejemplos implementación MATLAB\Segmentación fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Segmentada kmeans\','FileExtensions','.png', ...
    'IncludeSubfolders',true,'LabelSource','foldernames');

% DATOS PARA K-FOLD CROSS-VALIDATION
% Se lee el valor correspondiente a la entrada in_Kfold del programa
% principal
K_Folds = in_Kfold;
% Se asigna el campo .Labels de la estructura imds2 a etiq_imds2g
etiq_imds2g=imds2.Labels;
% Se crea la partición para la validación cruzada de los datos
cvp = cvpartition(etiq_imds2g, 'KFold', K_Folds);
Predictores_validaciong = NaN(size(etiq_imds2g));
num_observaciones = size(etiq_imds2g, 1);
num_clases = 2;
Puntajes_validaciong = NaN(num_observaciones, num_clases);

% K-FOLD CROSS-VALIDATION
for fold = 1:K_Folds
    % OBTENCIÓN DE LAS ETIQUETAS DEL ENTRENAMIENTO
    idx_fold = cvp.training(fold);
    imds2_entrenamiento_fold = imageDatastore('C:\Users\Jeferson\Desktop\Tesis\Computer vision cacao\Ejemplos implementación MATLAB\Segmentación fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Segmentada kmeans\','FileExtensions','.png', ...
        'IncludeSubfolders',true,'LabelSource','foldernames');
    etiquetas_entrenamiento_fold = imds2_entrenamiento_fold.Labels;
    imds2_entrenamiento_fold.Files = {imds2_entrenamiento_fold.Files{cvp.training(fold)}}';
    imds2_entrenamiento_fold.Labels = etiquetas_entrenamiento_fold(cvp.training(fold));
    etiq_imds2_entrenamientog = imds2_entrenamiento_fold.Labels;
```

```
33
34      % APLICACIÓN DEL OBJETO BOLSA DE PALABRAS VISUALES (BoVW)
35      % Se construye la bolsa de palabras visuales para la extracción de
36      % características y personalizando sus propiedades
37      Bolsa1 = bagOfFeatures (imds2_entrenamiento_fold, ...
38          'VocabularySize', 512, 'StrongestFeatures', 0.8, ...
39          'PointSelection','Detector');
40      % Se codifican las imágenes del datastore para el entrenamiento
41      Mat_caract_entrenamiento = encode(Bolsa1, …
            imds2_entrenamiento_fold);
42
43      % CLASIFICADOR SVM GAUSSIANO
44      imds2_entrenamientog = imds2_entrenamiento_fold.Labels;
45       clasificacion_SVMg = fitcsvm(...
46       Mat_caract_entrenamiento, ...
47       imds2_entrenamientog, ...
48       'KernelFunction', 'gaussian', ...                          % Se …
            especifica la función kernel como gaussiana
49       'OptimizeHyperparameters', 'auto', ...
50       'Standardize', true, ...
51       'ClassNames', categorical({'EXCELENTE CALIDAD'; 'MALA …
            CALIDAD'}));
52  [Puntajes_SVM_Clasificadorg,Parametros_puntajesg] = …
        fitSVMPosterior(clasificacion_SVMg);
53
54      % Se crea la estructura de resultados a través de la función …
            'predict'
55      svmPredictFcn = @(x) predict(Puntajes_SVM_Clasificadorg, x);
56
57      % OBTENCIÓN DE LAS ETIQUETAS PARA LA PRUEBA
58      imds2_prueba_fold = …
            imageDatastore('C:\Users\Jeferson\Desktop\Tesis\Computer …
            vision cacao\Ejemplos implementación MATLAB\Segmentación …
            fuzzy c-means\FRFCM\FRFCM PAR MEJOR\Segmentada …
            kmeans\','FileExtensions','.png', ...
59           'IncludeSubfolders',true,'LabelSource','foldernames');
60      etiquetas_prueba_fold = imds2_prueba_fold.Labels;
61      imds2_prueba_fold.Files = …
            {imds2_prueba_fold.Files{cvp.test(fold)}}';
62      imds2_prueba_fold.Labels = etiquetas_prueba_fold(cvp.test(fold));
63      % Se codifican las imágenes del datastore para la prueba
64      Mat_caract_prueba = encode(Bolsa1,imds2_prueba_fold);
65
66      % Se calculan las validaciones de predicción
67      [Predicciones_foldg, Puntajes_foldg] = …
            svmPredictFcn(Mat_caract_prueba);
68      % Se guardan las predicciones
```

```
69      Predictores_validaciong(cvp.test(fold), :) = Predicciones_foldg;
70      Puntajes_validaciong(cvp.test(fold), :) = Puntajes_foldg;
71  end
```

ORDEN DE EMPASTADO