PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DIAGNÓSTICO AUTOMÁTICO DE ROYA AMARILLA EN HOJAS DE CAFETO APLICANDO TÉCNICAS DE PROCESAMIENTO DE IMÁGENES Y APRENDIZAJE DE MÁQUINA

Tesis para optar el Título de **Ingeniero Informático**, que presentan los bachilleres:

Alfonso Carlos Barriga Pozada

Carlos Arrasco Ordoñez

ASESOR: Dr. César Beltrán Castañón

Lima, febrero de 2018

AGRADECIMIENTOS

A mi familia por apoyarme siempre en cada momento de mi vida.

A mi tía Meche y mi tío Dany por confiar en mí.

(Alfonso Barriga)

A Dios, El Señor Jesucristo, por darme la vida, su gracia, fuerza y provisión.
A mis padres Carlos y Carmen por su paciencia, apoyo, y respaldo en amor, consejo y oración.

A Alejandra, mi hermana, por sus consejos, oraciones y aliento en el proceso.

A Karis Paiva, mi prometida, y Familia por su ánimo, oración, alegría.

A mis Abuelitas Bertha y Angélica, por sus oraciones, y paciencia.

A los hermanos de la Iglesia Getsemaní por su Amor, y respaldo en oración.

A nuestro asesor César Beltrán por su apoyo.

(Carlos Arrasco)

RESUMEN

Actualmente, el café es uno de los recursos naturales más consumidos tanto en el mundo como en el Perú, Por ello, es menester garantizar la calidad en los granos de café, pues esto afectará considerablemente en el precio y posicionamiento en mercados altamente competentes; asimismo, el cultivo de este representa el principal ingreso para algunas familias, el cual se ve amenazado entre otras plagas, por la más perniciosa: La Roya Amarilla.

La Roya Amarilla se propaga fácilmente a través del aire, una vez que cae en un cultivo de café, ataca directamente en las hojas, almacenándose en forma de esporas en el envés de estas, y al paso de días consume las hojas hasta defoliar completamente la planta infectada. Debido a ello, la planta no puede adquirir los nutrientes necesarios del sol, pues necesita las hojas como receptores; en consecuencia, el fruto del café (granos) no se desarrollan con normalidad, y por ende su calidad y cantidad de cosecha es baja. Aun cuando no existe una solución absoluta para la erradicación de esta plaga, se la puede controlar; es decir, a través de un proceso manual y exhaustivo los caficultores pueden aplicar una solución bioquímica en la planta que detenga el desarrollo del hongo en las hojas, pero no acaba con ellas, solo se puede prolongar el tiempo de vida de la planta de café. Esto es posible, solo si se detecta en sus inicios la presencia de las esporas en las hojas, pues de haber germinado el hongo sería en vano cualquier intento de recuperar la planta, con lo que solo quedaría el exterminio de la planta.

Frente a este panorama, se propone una solución a través del aprendizaje máquina y procesamiento de imágenes, con el fin de automatizar el proceso de detección de la Roya en las hojas y calcular de manera más precisa la severidad del hongo. El proceso comienza en tomar fotografías a las hojas en un espacio semi controlado (con fondo blanco), luego se guardan todas las imágenes de las que se quiera conocer el porcentaje de severidad y ejecutar el programa propuesto, al término de ello el software muestra un reporte estadístico con el grado de incidencia por hoja según la clasificación de severidad que corresponda.

Finalmente, destacar que, de manera funcional, el aprendizaje máquina será vital para descartar si hay presencia de roya en la hoja analizada, y luego si la hoja está infectada, con el método de procesamiento de imágenes se calculará de manera más precisa el porcentaje de severidad considerando el área de la hoja examinada.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. GENERALIDADES

El presente capítulo se encuentra dividido en cinco secciones, las cuales serán descritas a continuación.

1.1 Problemática

La Roya Amarilla, es una de las enfermedades más peligrosas y serias que ataca a los cultivos del cafeto, siendo la más conocida y de peor reputación de entre todas las enfermedades de las plantas tropicales (Rayner, 1972). No solo afecta la cantidad de producción, sino la calidad del cultivo, lo que se traduce en millonarias pérdidas económicas y mermas de miles de puestos de trabajo (Vásquez. 2013). La Tabla 1.1 muestra la cantidad de puestos de empleo perdidos, a causa de la enfermedad, en los principales países cafetaleros.

Tabla 1.1 Referencial de pérdida de empleos según país en la región centroamericana. Año cafetero 2012-2013. Fuente: PROMECAFE

País	Número de empleos perdidos por la Roya (Año cafetero Octubre 2012 - Setiembre 2013)
Guatemala	115,000
Honduras	100,000
El Salvador	90,000
República Dominicana	56,500
Nicaragua	32,000
Panamá	30,000
Costa Rica	14,000
Jamaica	3,640

La enfermedad fue reportada por primera vez en 1869 en una isla asiática llamada Ceilán, actualmente conocida como Sri-Lanka (Avelino et al., 1999). Ese mismo año, el micólogo británico Berkeley se encargó de nombrar y describir al hongo causante de la enfermedad, Hemileia vastatrix (Avelino et al., 1999). Veinte años después del ataque de la Roya, Ceilán paso de ser el primer productor mundial de café a dejar de producirlo,

reemplazándolo principalmente por el cultivo de té, el cual se convirtió en la nueva bebida bandera del Imperio Británico (Rayner, 1972).

A partir de Ceilán, la Roya se expandió rápidamente por toda Asia y África, y en 1970 se detecta por primera vez en América, en el estado de Bahía, Brasil. La llegada de la Roya estremeció a los países productores, dado que en Latinoamérica todas las variedades cultivadas eran susceptibles, razón por la cual, en menos de veinte años, la Roya se hizo presente en todos los países productores de café (Avelino et al., 1999). En el Perú, la Roya fue reportada por primera vez en 1979, en la selva central, específicamente en la localidad de Satipo (Avelino et al., 1999).

El cultivo de café es considerado como el producto agrícola más importante en el comercio internacional, y una mínima reducción en el rendimiento o un ligero aumento en los costos de producción de este cultivo por efecto de la Roya Amarilla, puede tener un gran impacto en los cafeticultores y en los países cuyas economías son totalmente dependientes de las exportaciones del café (APS, 2011). En Brasil, por ejemplo, la carencia de medidas de manejo y control del patógeno mencionado condujo a una disminución del 30% en el rendimiento (Monaco, 1977). No obstante, la implementación de una calendarización de aplicaciones de fungicidas en las zonas cafetaleras de este país tuvo un costo de 67 US$/ha o 74 US$/m, lo cual representa el 9% del valor de las exportaciones de café de esta nación (CABI, 2013).

Este hongo alberga a la planta hasta su destrucción, y hasta ahora no es posible erradicarlo; es por ello, que los caficultores y agrónomos han buscado orientar la solución por la convivencia con la Roya, en vez de su exterminio, a través del control y buenas prácticas de cultivo. Para ello es menester conocer el grado de desarrollo de la Roya en la planta del café y así aplicar el tratamiento más eficiente, de lo contrario queda solo el exterminio de la planta.

Por otro lado, la aplicación de fungicidas y pesticidas sin ningún control o sin conocer la etapa en la que se encuentra la enfermedad puede traer como consecuencia serios daños al ecosistema. Normalmente, la detección del grado de severidad en hojas infectadas es realizada por la mera observación de los caficultores, que se basan en su experiencia para determinar qué acción tomar, lo cual contribuye a tener muchos errores de precisión y a la mala aplicación de las cantidades de pesticidas (Hitimana & Gwun, 2014).

En otras palabras, el problema central al cual se enfrentan los caficultores es la falta de herramientas que permitan una determinación precisa del grado de severidad de la Roya. Sin embargo, las marcadas características sintomatológicas, como las manchas amarillentas o la textura rugosa del hongo, permiten que el hecho de pensar en la aplicación de soluciones tecnológicas que detecten la enfermedad a través de la captura de imágenes y su posterior evaluación, sea algo factible de realizar.

Con el fin de buscar una solución a esta problemática, el Grupo de Reconocimiento de Patrones e Inteligencia Artificial Aplicada de la Pontificia Universidad Católica del Perú (GRPIAA - PUCP) presentó al Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica – CONCYTEC un proyecto que abarca desde la toma de la imagen en campo, hasta la evaluación de la misma con un dispositivo móvil, el cual resultó ganador de una subvención para su desarrollo. El presente trabajo forma parte de ese proyecto y tiene como finalidad desarrollar un modelo algorítmico, que a través de métodos de procesamiento de imágenes y aprendizaje de máquina permitiese detectar si una hoja está infectada o no con Roya Amarilla, y en caso lo estuviese, cuantificar de forma automática el grado de severidad. El alcance del trabajo no contempla la separación del fondo de la imagen y la hoja, se asume que las hojas llegan segmentadas y además no contempla el desarrollo de una aplicación móvil; sin embargo, el modelo algorítmico resultante será integrado en una aplicación de escritorio a manera de prueba.

1.1.1 Objetivo General

Desarrollar un modelo algorítmico que permita automatizar el proceso de diagnóstico de Roya Amarilla en hojas de cafeto, de tal forma que detecte cuando una hoja está o no infectada, y en caso lo estuviese, calcular el grado de severidad del hongo.

1.1.2 Objetivos Específicos

Los objetivos específicos son los siguientes:

1. Formar un banco imágenes de hojas de cafeto sanas e infectadas.
2. Implementar y parametrizar descriptores de imágenes que permitan extraer y generar vectores de características tanto de textura como de color de las imágenes analizadas.
3. Entrenar clasificadores, a partir de las características extraídas, que permitan detectar si una hoja está o no infectada.

4. Validar los clasificadores entrenados y determinar cuál de ellos obtuvo mejores resultados.
5. Aplicar un algoritmo de Clustering que permita segmentar las imágenes en áreas con Roya y sin Roya.
6. Aplicar un algoritmo de Thresholding que permita la segmentación de las imágenes en áreas con y sin Roya.
7. Cuantificar el porcentaje de avance de la enfermedad y determinar con cuál de los dos métodos de segmentación se obtienen mejores resultados.
8. Implementar el modelo algorítmico desarrollado en una aplicación de escritorio, para computadora, que reciba como entrada la imagen de una hoja de cafeto y dé como resultado un grado en la escala de severidad propuesta por la SAGARPA (Secretaría de Agricultura, Ganadería, Desarrollo Rural, Pesca y Alimentación).

1.1.3 Resultados esperados

Los resultados esperados son los siguientes:

1. *Relacionado con el objetivo específico 1*: Contar con un repositorio de imágenes estandarizadas, tomadas en un ambiente controlado, de al menos cincuenta hojas de cafeto infectadas con Roya Amarilla.
2. *Relacionado con el objetivo específico 2:* Script en Python que implemente descriptores tanto de textura como de color y que permita generar un vector de características de las imágenes.
3. *Relacionado con objetivo específico 3:* Modelos de clasificación que permitan detectar si una hoja tiene o no Roya Amarilla con al menos un 80% de precisión.
4. *Relacionado con el objetivo específico 4:* Reportes de métricas más utilizadas en el aprendizaje automático, determinación del descriptor con el que se obtuvo mayor precisión.
5. *Relacionado con los objetivos específicos 5:* Script en Python que implemente un método de Clustering y dé como resultado la imagen de la hoja segmentada en zonas con Roya y sin Roya.
6. *Relacionado con los objetivos específicos 6:* Script en Python que implemente un método de Thresholding y dé como resultado la imagen de la hoja segmentada en zonas con Roya y sin Roya.

7. *Relacionado con objetivo específico 7:* Script en Python que permita calcular el porcentaje de avance de la enfermedad según el área foliar afectada.
8. *Relacionado con objetivo específico 8:* Prototipo de aplicación que implemente el modelo algorítmico desarrollado.

El desarrollo de los objetivos se encuentra expresado en los capítulos 5, 6 y 7 del presente documento. La Tabla 1.2 muestra el mapeo de los objetivos con los respectivos capítulos.

Tabla 1.2 Mapeo de los objetivos y los capítulos del documento.

Capitulo	Objetivo especifico
5. Caracterización de regiones sanas e infectadas	Formar un banco imágenes de hojas de cafeto sanas e infectadas.
	Implementar y parametrizar descriptores de imágenes que permitan extraer y generar vectores de características tanto de textura como de color de las imágenes analizadas.
6. Clasificación automática de regiones sanas e infectadas.	Entrenar clasificadores, a partir de las características extraídas, que permitan detectar si una hoja está o no infectada.
	Validar los clasificadores entrenados y determinar cuál de ellos obtuvo mejores resultados.
7. Determinación automática del grado de severidad.	Aplicar un algoritmo de Clustering que permita segmentar las imágenes en áreas con y sin Roya.
	Aplicar un algoritmo de Thresholding que permita la segmentación de las imágenes en áreas con y sin Roya.
	Cuantificar el porcentaje de avance de la enfermedad y determinar con cuál de los dos métodos de segmentación se obtienen mejores resultados.

1.2 Métodos y procedimientos

A continuación se describen cada uno de los métodos y procedimientos utilizados en el proyecto.

1.2.1 Patrones Binarios Locales (LBP)

El operador LBP originalmente trabajaba con una zona de 3x3 (en realidad, este valor puede ser variable, dependiendo del tamaño de radio de los vecinos respecto al píxel del centro); es decir, un área de análisis considerando el valor umbral del píxel del centro como el principal sobre el cual se realizarán las operaciones de comparación respecto a la vecindad de los valores umbrales de píxeles contiguos (Porebski et al., 2008). Por ejemplo en la Figura 1.1 se tiene un operador básico LBP de 3x3:

56	15	47
34	**34**	10
23	58	12

Figura 1.1 Operador Básico LBP de 3x3.

Luego se compara cada valor de píxeles circundantes con el píxel central, quedando un patrón binario, el cual se muestra en la Figura 1.2:

1	0	1
1		0
0	1	0

Figura 1.2 Patrón binario.

En donde cada valor de píxel si es mayor o igual que el número de píxel central entonces se colocará un **1**, caso contrario, si es menor, se colocará un **0**.

De manera aleatoria para nuestro caso, se tomó la esquina superior izquierda en sentido horario, con lo cual se tiene un arreglo igual a: **10100101** números que en sistema decimal sería equivalente a **165**. Una vez que se ha aplicado la misma regla toda la imagen (recorrido de píxeles), se procede a dividir la imagen en bloques, cuyos tamaños pueden ser especificados; luego se calcula un histograma por cada uno de los bloques mencionado. Después, de realizados todos los histogramas, se procede con la

ecualización del número de píxeles en cada bloque y finalmente se juntan para obtener el vector de características resultantes de la imagen analizada.

Lo más relevante de las características LBP son la tolerancia frente a los cambios de iluminación monótona y la simplicidad computacional.

1.2.2 Descriptor de textura de Haralick

(E. Miyamoto y T. Merryman Jr., 2008) Se puede definir como un conjunto de las características de textura que son extraídas por medio de un método llamado Haralick, el cual define 14 funciones que muestran la relación entre el valor umbral de los píxeles de una imagen según la siguiente manera:

- Se selecciona cada píxel, y tomando como referencia a éste se toman a píxeles contiguos que se encuentren en la posición que marque el ángulo que hay entre el píxel central y estos, por ejemplo se evalúan la relación entre el píxel del medio y los que se encuentren alineados a los ángulos que forman con este de la siguiente manera: 0°, 45°, 90° 135° como se muestra en la Figura 1.3:

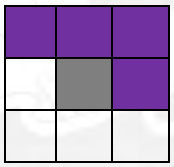

Figura 1.3 Vecindario de píxeles válidos

En donde el recuadro central representa el píxel evaluado y los que están de color gris representan los píxeles alineados con los ángulos anteriormente mencionados.

- Luego, se procede a aplicar 14 funciones que define Haralick tomando como datos de entrada la relación entre el píxel central y los contiguos seleccionados (en gris), y aquello mismo se realiza con cada píxel acumulando los resultados parciales en cada variable que indicará finalmente cada característica extraída, 14 en total por imágenes.

  Las siguientes ecuaciones definen las características:

  $p(i,j)$: Es la entrada (i, j) en una dependencia espacial tono-gris con matriz normalizada = $P(i,j)/R$; R: Constante de Normalización

- Segundo momento angular:

$$f_1 = \sum_i \sum_j \{p(i,j)\}^2$$

- Contraste:

$$f_2 = \sum_{n=0}^{N_{g-1}} n^2 \{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \}$$

Donde: $|i - j| = n$

- Correlación:

$$f_3 = \frac{\sum_i \sum_j (ij) p(i,j) - u_x u_y}{\sigma_x \sigma_y}$$

Donde: u_x, u_y, σ_x y σ_x significan la desviación estándar de p_x y p_y

- Suma de cuadrados: Varianza

$$f_4 = \sum_i \sum_j (i-u)^2 p(i,j)$$

- Momento de diferencia inversa:

$$f_5 = \sum_i \sum_j \frac{1}{1 + (i-j)^2} p(i,j)$$

- Suma promedio:

$$f_6 = \sum_{i=2}^{2N_g} i p_{x+y}(i)$$

- Suma de varianza:

$$f_7 = \sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i)$$

- Suma de entropía:

$$f_8 = \sum_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\}$$

- Entropía:

$$f_9 = -\sum_i \sum_j p(i,j) \log(p(i,j))$$

- Diferencia de varianza:

$$f_{10} = varianza(p_{x-y})$$

- Diferencia de entropía:

$$f_{11} = \sum_{i=0}^{N_{g-1}} p_{x-y}(i) \log\{p_{x-y}(i)\}$$

- Medidas de información de correlación:

$$f_{12} = \frac{HXY - HXY1}{\max\{HX, HY\}}$$

$$f_{13} = (1 - \exp[-2.0(HXY2 - HXY)])^{1/2}$$

Dónde:

- $HXY = -\sum_i \sum_j p(i,j) \log(p(i,j))$
- HX y HY son entropías de p_x y p_y
- $HXY1 = -\sum_i \sum_j p(i,j) \log\{p_x(i) p_y(j)\}$

- $HXY2 = -\sum_i \sum_j p_x(i)p_y(j)\log\{p_x(i)p_y(j)\}$

- Coeficiente de máxima correlación:

$$f_{14} = (Segundo\ mayor\ valor\ propio\ de\ Q)^{1/2}$$

Donde:

- $(i,j) = \sum_k \frac{p(i,j)p(j,k)}{p_x(i)p_y(k)}$

1.2.3 Máquinas de Vectores de Soporte (SVM)

SVM (del inglés *Support Vector Machine*) es uno de los algoritmos de aprendizaje de máquina más populares en la actualidad (Hsu, Chang, & Lin, 2003). Fue introducido por primera vez por Vapnik, V. en 1992, y ha sido ampliamente usado desde la fecha, principalmente porque, en comparación con otros algoritmos de aprendizaje, ofrece una precisión superior en problemas de clasificación para conjuntos de datos de razonable tamaño (Marsland, 2009).

Para el problema de clasificación binaria, dado por dos clases, se tiene el siguiente modelo lineal:

$$y(x) = w^T \emptyset(x) + b$$

En el cual ∅(x) representa una transformación fija del espacio de características y b representa un parámetro de parcialidad (Bishop, 2006). El conjunto de datos de entrenamiento esta formador por N vectores de entrada x_1,… x_N con sus correspondientes valores objetivo, o etiquetas t_1,… t_N en el cual t_N ∈ {-1,1}, y los nuevos puntos son clasificados según el signo de y(x) (Bishop, 2006). En SVM, este problema es atacado a partir del concepto de margen, el cual es definido como la distancia mínima entre el límite de decisión y cualquiera de los puntos clasificados (Bishop, 2006). Los puntos en cada clase que fueron clasificados de manera más cerca al límite de decisión son conocidos como *Support vectors* (Marsland, 2009). Por lo tanto, el objetivo es conseguir un margen lo más largo posible y; además, tener en cuenta que los *Support vectors* tienen un rol principal en el método, lo cual deja entre ver una importante característica de este algoritmo, y es que luego de haberse realizado el

entrenamiento, todos los datos, excepto los *Support vectors*, pueden ser eliminados ya que no serán necesarios para la clasificación (Marsland, 2009).

1.2.4 K-means clustering

Es uno de los algoritmos más conocidos de Clustering que existe, fue desarrollado en 1967 por MacQueen. Su manera de clasificar elementos en grupos es bastante sencilla, pero eficaz en cuanto a aplicación refiere, este número de grupos es determinado previamente por la variable K (MacQueen, 1967).

Su nombre se deriva del hecho de que el algoritmo agrupa la media de los puntos que componen cada grupo respectivamente; es decir, los agrupa por la media ponderada de los centroides de las coordenadas de sus puntos. La ventaja de ello es que tiene un significado estadístico y gráfico de manera inmediata. Para llevar a cabo el desarrollo de este algoritmo, se deben seguir los siguientes 4 etapas (Cambronero, 2006):

1. Se eligen aleatoriamente el número de grupos (*clusters*) iniciales, con un centro inicial para cada objeto que compone el grupo igual a su coordenada en el eje x.
2. Luego, se reasignan los objetos de los grupos, con el criterio que la clasificación es el más cercano al objeto, según la media de sus distancias.
3. Después, de que todos los objetos son reasignados se vuelve a calcular los centros de cada grupo (los baricentros).
4. Repetir el paso 2 y 3 hasta que no se hagan más reasignaciones.

Para minimizar el efecto de la elección aleatoria, se debe probar muchas veces el algoritmo sobre el mismo conjunto de datos, sabiendo que a centros más espaciados se encuentran mejores resultados.

1.2.5 Thresholding

Se denomina *Thresholding* o método umbral a la operación en la cual se define un rango de valores de intensidad en la imagen original, se seleccionan los pixeles pertenecientes a este rango como si fueran pertenecientes al *foreground* (primer plano), o región a separar, y se rechazan todos los demás como si fueran parte del *background* (fondo). Dicha imagen es usualmente mostrada como una imagen binaria o de dos niveles, usando blanco y negro para distinguir las regiones (Russ, 1999).

Por ejemplo, dada una imagen *f(x,y),* compuesta por objetos brillantes en un fondo oscuro, los pixeles se dividirán en dos grupos según su intensidad. La manera de separar el fondo de los objetos es seleccionar un valor umbral (*threshold*), *T*, de tal manera que la imagen quede definida de la siguiente manera:

$$g(x,y) = \begin{cases} 1 \; si \; f(x,y) > T \\ 0 \; si \; f(x,y) \leq T \end{cases}$$

Cuanto *T* es constante para toda la imagen, el proceso es denominado *global Thresholding*. Cuando *T* varía a lo largo de la imagen, el proceso es conocido como *variable Thresholding.* Por otro lado, cuando una imagen presenta más de dos modos se subdividirse, por ejemplo una en la cual los objetos brillantes en el fondo oscuro son de un grado de intensidad diferente, se definen múltiples valores umbral y el proceso se denomina *múltiple Thresholding* (Gonzales & Woods, 2008).

En la Figura 1.4 se observa un ejemplo básico de una imagen antes y después de haberle aplicado un *global Thresholding.*

Figura 1.4 Tomada de la documentación de OpenCV

1.2.6 Método Otsu

Este método es muy importante debido a que se necesita conocer el valor umbral de una imagen en gris para poder aplicar el método Thresholding, este método brindará el valor umbral óptimo para cada imagen (Otsu, 1979) . Para la consecución de ello, es necesario emplear las siguientes fórmulas:

- Umbral óptimo:

$$T = \max(\sigma^2)$$

Donde:

- Varianza:

$$\sigma^2 = w_B(\mu_B - \mu)^2 + w_F(\mu_F - \mu)^2$$

- Probabilidad Acumulada:

$$w_k = \sum_{i=0}^{k} p_i$$

- Media Acumulada:

$$\mu_k = \sum_{i=0}^{k} i.p_i$$

- Probabilidad Acumulada:

$$\mu_c = \frac{\mu_k}{\omega_k}$$

B: Corresponde al fondo de la imagen ("*Background")*

F: Corresponde al objeto de la imagen ("*Foreground")*

De esta manera, el valor del umbral óptimo está expresado por T (N. Otsu, 1979).

1.3 Herramientas

A continuación se describen las herramientas que tuvieron más relevancia en el desarrollo del proyecto.

1.3.1 OpenCV (Open Source Computer Vision Library)

OpenCV es una librería o paquete de código abierto bajo la licencia BSD, la cual alberga cientos de algoritmos de visión computacional. Fue diseñada para ser muy eficiente, ya que está escrita en C/C++ optimizado y toma ventaja del multiprocesamiento. Actualmente, cuenta con una comunidad de usuarios de más de 47 mil personas, ha

sido adoptada y ampliamente utilizada alrededor del mundo, ya que soporta plataformas Linux, Windows, Mac OS, IOS, Android; y además, brinda interfaces para Java, Python, C y C++.

1.3.2 Scikit-Learn: Machine Learning in Python Library

Scikit-Learn es un módulo o librería para Python que integra una gran cantidad de algoritmos de aprendizaje de máquina. El objetivo del paquete es brindar a las personas, que no son especialistas en el tema, la capacidad de reproducir los más avanzados algoritmos (Pedregosa et al., 2011). Entre sus principales características se encuentran:

- Simple y eficiente herramienta utilizada en minería y análisis de datos.
- Accesible a cualquier persona, reusable en varios contextos.
- Trae consigo integradas otras tres damita.
- Es de código abierto y puede usarse para fines comerciales.

1.3.3 Anaconda Scientific Python Distribution

Es una colección poderosa de paquetes para Python que permite la gestión a gran escala de datos, análisis y visualización para la Inteligencia de Negocios, Análisis Científico e Ingeniería de aprendizaje automático, también provee un entorno de desarrollo (IDE) de distribución libre para Python con el cual se ha desarrollado la programación en este proyecto. Este IDE contiene más de 195 paquetes (librerías) para este lenguaje de programación.

Anaconda provee instaladores para Python 2.7 y 3.4. Para efectos del desarrollo se optó por Python 2.7.

1.4 Alcance

Esta investigación aplicada busca automatizar y mejorar la precisión de los resultados en el diagnóstico de Roya Amarilla en hojas de cafeto, considerando un alto índice de certeza, el cual será corroborado por el análisis de un experto en el tema, idealmente, o por contraste con un etiquetado de forma manual.

Finalmente, se aclara que las fotos serán obtenidas de un ambiente controlado, no forma parte del estudio la extracción de un *background* complicado, como seria en el caso de que la foto fuera tomada en los mismos campos de cultivo.

1.4.1 Limitaciones

El diseño de la investigación exige el manejo de fotos con una resolución y tamaño determinados para la evaluación en el procesamiento de imágenes; por lo que contar con imágenes de menor calidad y dimensiones inferiores producirá errores en el vector resultante de la foto. Asimismo, la iluminación de la foto es un factor crítico, pues si la imagen es muy opaca, debido a sombras, o muy iluminada, como consecuencia de un día soleado, los métodos podrían arrojar resultados errados.

1.4.2 Riesgos

En la Tabla 1.3 se muestran los riegos que pudieron afectar el correcto desarrollo del proyecto.

Tabla 1.3 Matriz de Riesgos

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
Pérdida de la información del proyecto.	Alto	Tener un respaldo actualizado en la nube informática, así también en dispositivos de almacenamiento externos y discos duros internos.
Privatización de herramientas de desarrollo informático que son actualmente libres y gratuitas.	Medio	Contar con herramientas suplentes que puedan realizar las mismas funciones.
Retiro de alguno de los recursos (alumnos) del proyecto.	Medio	Identificar las tareas por recurso y fechas de entrega de metas. Asimismo, manejar tiempos de holgura en el proyecto, en caso se tenga que aumentar la carga en un recurso.
Ausencia del asesor en etapas claves del proyecto (sean por viajes,	Medio	Mantener buena comunicación con el asesor del proyecto, y fechas de entrega para el

capacitaciones, o motivos personales).		cumplimiento de las metas; así como también, manejar medios de comunicación directos en caso de distanciamiento físico.
Adelanto de fecha de entrega del proyecto.	Medio	Durante la etapa de planificación de desarrollo del proyecto, contemplar tiempos de holgura.

1.5 Justificativa y viabilidad del proyecto

A continuación se presentan los motivos por los cuales se consideró esta investigación como justificada y su respectivo análisis de viabilidad.

1.5.1 Justificativa

La Roya Amarilla del café es una de las enfermedades tropicales que ocasiona mayores pérdidas económicas, dado que se expande rápidamente, es casi imposible erradicarla y no solo afecta la cantidad de cultivo sino también su calidad. Uno de los puntos débiles en la lucha para combatir y controlar esta plaga es la rapidez y precisión con la que es detectada y diagnosticada, ya que el proceso de evaluación es manual y se necesita de una elevada cantidad de personas que vayan por la plantación de café recopilando información acerca de las características de las hojas. Luego, esta información debe ser tabulada y procesada para tomar una decisión sobre el curso de acción a seguir. Sin embargo; el hecho de que la recopilación de características sea manual deja mucho margen de error, por lo cual no se estima bien el avance de la enfermedad y se opta por cantidades elevadas, o muy bajas, de fungicidas y pesticidas para combatirla.

El presente proyecto ataca directamente esta falencia, ya que busca automatizar el proceso detección y diagnóstico a través del uso de algoritmos de procesamiento de imágenes y aprendizaje automático. Con la herramienta final, los productores de café podrán acelerar el proceso de detección y conocer con mayor rapidez el grado de avance de la enfermedad. El proceso dejaría de ser manual y no se necesitaría que el caficultor sea un experto en Roya Amarilla, puesto que la herramienta estará en capacidad de reconocer las características de la enfermedad sin mayor información que las fotos de las hojas de cafeto. Por otro lado, el alto índice de precisión de la herramienta permitirá tomar mejores decisiones acerca del uso de pesticidas y fungicidas, lo cual a su vez

disminuirá el impacto en el ecosistema, ocasionado muchas veces por un uso indiscriminado y sin sustento de estos químicos.

El impacto económico sería significativo, en primer lugar por el ahorro de recursos destinados a la detección y, en segundo lugar, por la mejor asignación de recursos destinados a la compra de fungicidas y pesticidas, ya que se conocerá con mayor precisión el grado de severidad. Los caficultores de pocos recursos, que no pueden costear expertos o largos procesos de evaluación, se verían beneficiados puesto que el uso de la herramienta no tendría costo alguno y lo máximo requerido son las fotos de sus plantaciones. La herramienta, además, al ser capaz de diferenciar los síntomas y características de la plaga, servirá como una fuente de conocimiento a los caficultores no expertos.

En un futuro, el modelo algorítmico desarrollado se podría integrar en aviones no tripulados o popularmente llamados D*rones*, con el fin de lograr un monitoreo en tiempo real de las plantaciones de café. El modelo también se puede integrar con un sistema en la nube a través del cual, caficultores de todo el mundo, envíen a través de la web las imágenes de hojas de cafeto de sus plantaciones y el diagnostico sea procesado en servidores externos.

1.5.2 Análisis de viabilidad

La presente investigación aplicada resulta viable económicamente puesto que todas las herramientas de software a utilizarse serán código libre, no se requiere de mayores implementos salvo una computadora de escritorio o laptop. Además, la investigación forma parte del proyecto llamado “Análisis automático mediante procesamiento de imágenes digitales para determinar el grado de severidad de la Roya Amarilla en hojas de cafeto”, el cual pertenece al Grupo de Reconocimiento de Patrones e Inteligencia Artificial Aplicada (GRPIAA) que ganó el financiamiento del CONCYTEC para su desarrollo.

2 MARCO CONCEPTUAL

El presente capítulo se encuentra dividido en tres secciones, las cuales serán desarrolladas a continuación.

2.1 La Roya Amarilla del café

En los siguientes puntos se explica teoría necesaria para entender el ciclo de vida la Roya Amarilla, su sintomatología y métodos de evaluación y tratamiento.

2.1.1 Clasificación taxonómica y hospederos

La Roya Amarilla del Cafeto es una enfermedad causada por el hongo Hemileia Vastatrix, su principal forma de multiplicación es a través de las uredosporas, que son esporas unicelulares, o cuerpos reproductivos, característicos de este tipo de hongo (Avelino et al., 1999). La clasificación taxonómica del hongo se muestra en la tabla 2.1.

Tabla 2.1 Clasificación Taxonómica - Tomada de (Avelino et al., 1999)

Clase:	*Basidiomycetes*
Subclase	*Teliomycetidae*
Orden:	*Uredinales*
Familia:	*Pucciniaceae*
Género:	*Hemileia*
Especie:	*Vastatrix*

2.1.2 Sintomatología

Los síntomas de la enfermedad se manifiestan inicialmente como pequeñas lesiones o manchas redondas de color amarillo-anaranjado traslucido en el envés (parte posterior) de la hoja. Una sola mancha o lesión tiene un diámetro de 3 mm pero gradualmente aumenta el tamaño hasta 2 cm o más y puede unirse a otras para formar una lesión irregular que puede abarcar toda la hoja. Según Rayner (1972), "*Uno o dos días después de la primera aparición, la mancha toma un tinte anaranjado y la superficie se vuelve polvorienta y comienza la formación de esporas*". En infecciones avanzadas, si la esporulación es abundante, un solo golpe en la hoja puede levantar una nube de esporas. Las esporas son de tamaño microscópico de forma reniforme, lisa en la cara interna y rugosa en la cara externa, corresponden al polvillo de color amarillo o anaranjado que se observa en el envés de las hojas infectadas. "*Finalmente, cuando las*

áreas atacadas por el hongo se hacen más viejas, su centro muere, se vuelve marrón oscuro y se seca", (Rayner, 1972). Sin embargo, a pesar de que las lesiones viejas se necrosan, la esporulación puede continuar en el margen de la lesión.

En la Figura 2.1 se observa el desarrollo de los síntomas provocados por la Roya. En (A) están presentes manchas traslucidas, en (B) las manchas aumentan y se produce la esporulación y en (C) se observa una hoja con lesiones viejas. En la Figura 2.2 se observa, a la izquierda, una hoja con lesiones secas y, en la Figura 2.3, una hoja con abundante esporulación.

Figura 2.1 Tomada de Recomendaciones CICAFE

Figura 2.2 Tomada de Recomendaciones CICAFE

Figura 2.3 Tomada de UNSAAC

2.1.3 Metodología de evaluación y diagnóstico

En la investigación realizada por la Secretaría de Agricultura, Ganadería, Desarrollo Rural, Pesca y Alimentación (SAGARPA, 2013) de México, se propone usar escalas descriptivas y gráficas para medir la incidencia y la severidad de la Roya Amarilla. Primero evalúan el daño en la planta, tomando en cuenta el porcentaje foliar del cafeto afectado por la Roya y luego evalúan la severidad en las hojas. Para la primera parte utilizan una escala dividida en seis grados y para la segunda parte utilizan una escala dividida en cinco grados. En las Figuras 2.4 y 2.5 se observan las escalas utilizadas para cada parte de la evaluación.

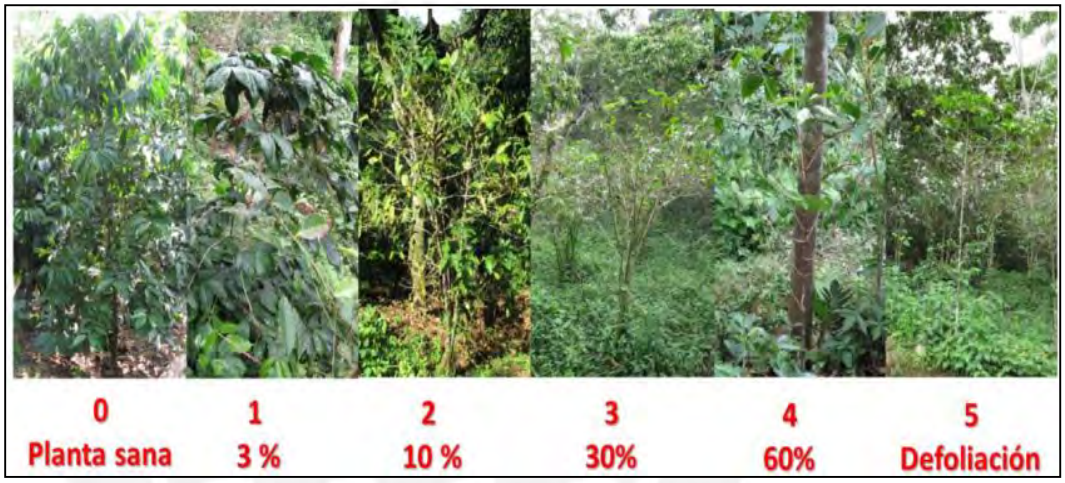

Figura 2.4 Escala de evaluación en la planta de cafeto, tomada de SAGARPA 2013

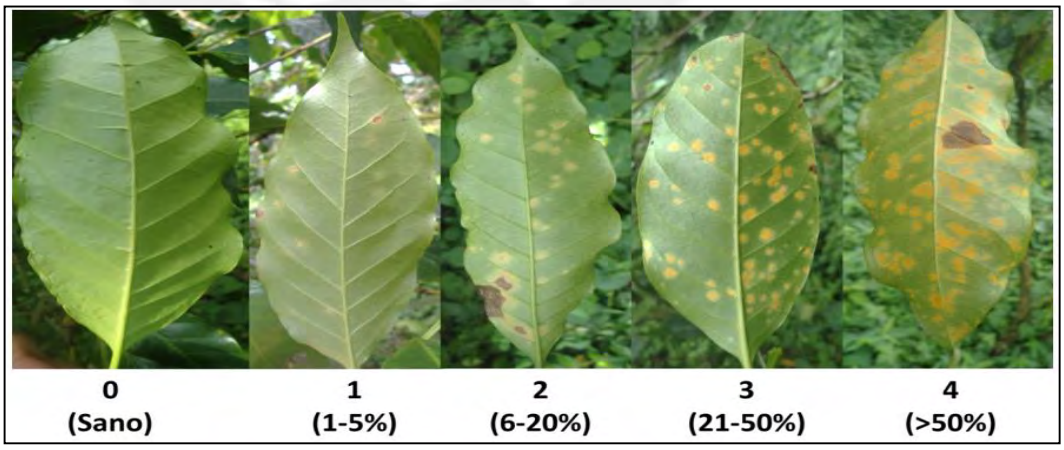

Figura 2.5 Escala de evaluación en las hojas, tomada de SAGARPA 2013

Finalmente, con los datos recolectados del trabajo de campo, se calculan los índices de daño en la planta y los índices de daño en las hojas.

2.2 Procesamiento de imágenes digitales

En los siguientes puntos se describen los principales conceptos acerca del procesamiento de imágenes, indispensables para entender el desarrollo de esta investigación.

2.2.1 Imagen digital

Una imagen digital puede ser definida como una función de dos dimensiones, f(x,y), donde "x" e "y" son coordenadas en el plano y la amplitud de "f" en cualquier punto (x,y) es conocida como la intensidad de la imagen en ese punto (Gonzales & Woods, 2008). Cuando los puntos (x, y) y el valor de la intensidad de f son finitos, o discretos, se conoce a la imagen como imagen digital. El procesamiento de imágenes hace referencia al uso de la computadora para la obtención de características, valores numéricos que permiten describir la imagen, teniendo en cuenta que una imagen digital es una colección finita de elementos con una posición particular, estos elementos normalmente son llamados pixeles (Gonzales & Woods, 2008).

2.2.2 Espacios de color

En líneas generales, los espacios de color son la especificación de un sistema de coordenadas y un sub-especio en el cual cada color es representado por un punto (Gonzales & Woods, 2008). Estos sistemas permiten facilitar la especificación de colores siguiendo un estándar, generalmente aceptado (Gonzales & Woods, 2008).

- **RGB(*Red, Green, Blue*)**

En el modelo RGB el valor de un color en particular es representado como un vector de tres elementos, que son la intensidad de los tres colores primarios (Sonka, Hlavac, & Boyle, 2008). El modelo está basado en el sistema cartesiano, en el cual el rojo, verde y azul representan cada uno de los ejes coordenados *x,y,z* (Gonzales & Woods, 2008). En la Figura 2.6 se observa el sub-espacio de colores

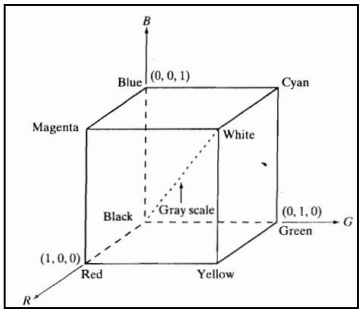

Figura 2.6 Sub-espacios de colores. Tomada de (Gonzales & Woods, 2008)

- **HSI (*Hue,Saturation, Intensity*)**

EL modelo HSI desacopla información acerca de la intensidad (*intensity*) a partir de los colores, mientras que, los tonos (*hue*) y la saturación (*saturation*) corresponden a la percepción humana. Por esta razón, el modelo HSI es adecuado para desarrollar algoritmos de procesamiento de imágenes que tomen en cuenta descriptores de color naturales a la visión humana (Gonzales & Woods, 2008).

- **CIE Luv y CIE Lab**

Ambos espacios de color fueron propuestos en 1976 por la Comisión Internacional de Iluminación (CIE) con el objetivo principal de proveer un espacio perceptualmente uniforme, lo cual quiere decir que la distancia Euclidiana entre dos colores del espacio está fuertemente relacionada con la percepción visual del ser humano(Tkalcic & Tasic, 2003). La principal diferencia entre CIE Luv y CIE Lab es el modelo de adaptación cromática implementado; el espacio CIE Lab normaliza sus valores dividiéndolos entre el valor del punto blanco, mientras que el espacio CIE Luv normaliza sus valores mediante la sustracción del punto blanco (Tkalcic & Tasic, 2003).

2.2.3 Segmentación

Sea R la región espacial ocupada por una imagen, el proceso de segmentación es aquel en el cual el espacio R es dividido en *n* subregiones, $R_1, R_2, R_3, \dots, R_n$, de tal forma que se cumplan las siguientes condiciones(Gonzales & Woods, 2008).:

a. $i = 1nR|i = R$
b. $R_i esunconjuntoconexo, i = 1,2, \dots, n$
c. $R_i R_j = \emptyset paratodoiyj, i \neq j$
d. $Q(R_i) = Vparai = 1,2, \dots, n$

e. $Q(R_i \cup R_j) = Fparacualquierregionadyacente R_i y R_j$

Los algoritmos de segmentación más conocidos se basan en propiedades básicas de la imagen, tales como la discontinuidad y la similitud (Gonzales & Woods, 2008). Con relación a la discontinuidad, el proceso se basa en dividir la imagen según cambios abruptos en rangos de intensidad (Gonzales & Woods, 2008). Con respecto a la segmentación, el proceso se basa en dividir la imagen en regiones similares de acuerdo a un conjunto de criterios predefinidos (Gonzales & Woods, 2008).

- **Segmentación basada en regiones**

Consiste en agrupar los pixeles o subregiones en regiones basadas en un criterio predefinido de crecimiento. Se inicia con un conjunto de pixeles llamados semillas, los cuales agruparan con otros pixeles vecinos según algún criterio predefinido, como rangos de intensidad o color. Si no se conoce información a priori acerca de la imagen, se puede aplicar alguna técnica de agrupamiento (*Clustering*) de tal modo que se evalúe cada pixel y se agrupen por similitud. Los pixeles más cercanos al centroide de un grupo o *clúster* encontrado podrían ser candidatos a convertirse en semillas. El crecimiento de una región se detiene cuando no existen más pixeles que satisfagan el criterio de inclusión en la región (Gonzales & Woods, 2008).

2.2.4 Descriptores de imágenes

Un descriptor es una operación que permite extraer características que mejor representen una imagen, por ello la salida es un vector de características numéricas, invariantes a transformaciones básicas en la imagen y que pueden ser comparados con vectores de otros descriptores en tareas como la clasificación de imágenes, reconocimiento de objetos y consulta de imágenes en bases de datos de características (Manjunath, Ohm, Vasudevan, & Yamada, 2001).

2.2.5 Operaciones morfológicas

Un operador morfológico es una herramienta que permite extraer componentes de una imagen que son útiles en la representación y descripción de la forma de una región (Gonzales & Woods, 2008). Las principales operaciones morfológicas son la Erosión y la Dilatación. Dado que ambas operaciones no son inversas una de la otra, la combinación de estas dan lugar distintas transformaciones, las más comunes son la

Apertura y el Cierre (Sonka et al., 2008). A continuación se describe dos de las técnicas utilizadas para el proyecto:

- **Erosión:**

Dados dos conjuntos *A* y *B* definidos en Z^2, la erosión de *A* dada por *B,* denotada como $A \ominus B$, se define como:

$$A \ominus B = \{ z|(B_z) \subseteq A \}$$

La ecuación tiene como significado que la erosión de *A* dada por *B* es el conjunto de puntos *z* tales que *B*, representado por *z*, está contenido en *A.* En otras palabras, esta operación encoge o adelgaza objetos en una imagen, la forma específica y el alcance del encogimiento son controlados por un objeto estructural (Gonzales & Woods, 2008).

- **Dilatación:**

Dados dos conjuntos *A* y *B* definidos en Z^2, la erosión de *A* dada por *B,* denotada como $A \oplus B$, se define como:

$$A \oplus B = \left\{ z|(\hat{B})_z \cap A \neq \emptyset \right\}$$

La dilatación de *A* dada por *B* es el conjunto de puntos *z,* desplazamientos, tal que *A* y *B* se superponen por al menos un elemento. En otras palabras, esta operación permite hacer crecer o engrosar objetos en una imagen, la forma específica y el alcance del crecimiento son controlados por un objeto estructural (Gonzales & Woods, 2008).

2.3 Aprendizaje máquina

El aprendizaje de máquina es una rama de la inteligencia artificial que tiene por objetivo desarrollar métodos para hacer que las computadoras modifiquen o adapten sus acciones de tal forma que dichas acciones se vuelvan más precisas, es decir, que puedan aprender y elegir acciones cada vez más cercanas a las correctas (Marsland, 2009). El aprendizaje de maquina es normalmente dividido en dos tipos: El aprendizaje supervisado y el aprendizaje no supervisado

2.3.1 Aprendizaje supervisado

Está basado en el aprendizaje sobre la experiencia. Lo conforman el conjunto de algoritmos y técnicas capaces de aprender a partir de datos de entrenamiento, o en otras palabras, ejemplos con la respuesta correcta. Estos algoritmos deberán ser capaces de generalizar la respuesta para cualquier input que no esté en los datos de entrenamiento (Marsland, 2009).

2.3.2 Aprendizaje no supervisado

El objetivo principal del aprendizaje no supervisado es encontrar, en un conjunto de datos de entrada, subgrupos con características similares sin tener la necesidad de tener información previa o datos de entrenamiento. A diferencia del aprendizaje supervisado, el algoritmo deberá hallar las similitudes por sí mismo y clasificar los elementos según las características que haya logrado identificar (Marsland, 2009).

2.3.3 Reconocimiento de patrones y clasificación

Un patrón es un arreglo o conjunto de características, las cuales, dentro del campo del análisis de imágenes, son conocidas como descriptores, de los cuales se habló en el apartado anterior. Una clase es una familia de patrones que comparten propiedades en común (Gonzales & Woods, 2008).

La clasificación dentro del área del aprendizaje automático hace referencia al problema de encontrar, para un elemento con determinadas características, la clase a la cual pertenece. Un punto importante dentro de la clasificación es que esta es discreta, un elemento corresponde únicamente a una clase, y el conjunto de clases cubren todo el posible espacio de salida (Marsland, 2009). Por ejemplo, en el presente trabajo, se implementara un método de clasificación con dos posibles clases, hojas con Roya Amarilla y hojas sanas.

2.3.4 Técnicas para la validación de modelos de clasificación

- **Cross-validation:**

La validación cruzada, es una técnica modelo de validación para evaluar cómo los resultados de un análisis estadístico serán generalizados a un conjunto de datos independientes. El contexto para su uso es la predicción que se quiere estimar mediante la práctica.

Este modelo generalmente administra un conjunto de datos llamados conocidos y otro conjunto de datos llamados desconocidos (o evaluados por primera vez), contra el cual el modelo se prueba (conjunto de datos de prueba).

El propósito de la validación cruzada es definir un conjunto de datos para probar el modelo en la fase de entrenamiento; es decir, el conjunto de datos de validación, y esto para restringir los problemas de sobreajuste. Así, se podrá conocer de manera generalizada una información desconocida independiente (Witten & Frank, 2011).

- **Leave One Out**:

Este procedimiento es atractivo por dos razones. En primer lugar, la mayor cantidad posible de datos que se utiliza para el entrenamiento, en cada caso, aumenta la posibilidad de que el clasificador sea más preciso. En segundo lugar, el procedimiento es determinista: es decir, no hay ningún punto en repetir el procedimiento 10 veces, o ir repitiéndolo en absoluto: ya que el mismo resultado se obtendrá cada vez.

La desventaja de ello es el alto costo computacional, ya que todo el proceso de aprendizaje debe ser ejecutado n veces y esto es por lo general imposible para grandes conjuntos de datos. Sin embargo, dejar uno fuera parece ofrecer una oportunidad de exprimir el máximo rendimiento de un pequeño conjunto de datos y llegar como una estimación precisa como sea posible (Witten & Frank, 2011).

2.3.5 Métricas de evaluación del rendimiento de un clasificador

Un punto sumamente importante en el aprendizaje de máquina es la evaluación del desempeño del modelo de clasificación, para lo cual existen distintas medidas de rendimiento que no necesitan lidiar con la estructura interna del algoritmo. A continuación, dadas dos clases, una positiva y otra negativa, se describen las principales métricas utilizadas para evaluar el desempeño de un clasificador binario (Powers, 2011):

- ***True Positives* (TP):** Número de casos positivos a los cuales se les asignó la clase positiva.
- ***True Negatives* (TN)**: Número de casos negativos a los cuales se les asignó la clase negativa.
- ***False Positives* (FP):** Número de casos negativos a los cuales se les asignó la clase positiva.

- ***False Negatives* (FN):** Número de casos positivos a los cuales se les asignó la clase negativa.
- ***Accuracy (ACC)*:** Número de casos clasificados correctamente respecto del total de casos.

$$ACC = (TP + TN)/(TP + TN + FP + FN)$$

- ***Error Rate (ER)*:** Número de casos clasificados incorrectamente respecto del total de casos.

$$ER = (FP + FN)/(TP + TN + FP + FN)$$

- ***Recall - Sensitivity (TPR)*:** Esta métrica, también llamada TPR (del inglés *True Positive Rate*), mide la proporción de casos reales positivos que han sido clasificados correctamente como casos positivos.

$$TPR = TP/(TP + FN)$$

- ***Precisión (TPA)*:** Esta métrica, también llamada TPA (del inglés *True Positive Accuracy*), mide la proporción de casos que han sido clasificados como positivos que son realmente casos positivos.

$$TPA = TP/(TP + FP)$$

- ***F-Measure:*** Esta métrica es la media armonica entre las métricas de Recall y Precision.

$$F_{MEASURE} = 2 * (TPR * TPA)/(TPR + TPA)$$

- **Matriz de confusión**

La exactitud de un clasificador binario puede ser evaluada calculando el número de muestras a las cuales se les asignó su clase correctamente (True Positives), el número de muestras en las que se reconoció correctamente que no pertenecían a la clase (True Negatives), el número de muestras a las cuales se les asignó incorrectamente la clase (False Positives) y el número de muestras que, perteneciendo a la clase, esta no les fue asignada (False Negatives); estas cuatro frecuencias o conteos constituyen la matriz de confusión (Sokolova & Lapalme, 2009). A continuación se muestra la Matriz de confusión en la Tabla 2.2:

Tabla 2.2 Modelo de Matriz de confusión (Kohl, 2012).

		Predicción	
		Clase Negativa (-1)	Clase Positiva (+1)
Realidad	Clase Negativa (-1)	True Negative (TN)	False Positives (FP)
	Clase Positiva (+1)	False Negatives (FN)	True Positives (TP)

3 ESTADO DEL ARTE

En el presente capítulo se realizó un resumen de los principales trabajos de investigación y avances relacionados al procesamiento de imágenes y clasificación de enfermedades en plantas. Se han tomado en cuenta distintas investigaciones modernas que aplican nuevas o mejoradas técnicas de segmentación, detección y clasificación.

3.1 Avances en la caracterización de regiones en base al color

En la investigación realizada en (Wei-Ta, Wei-Chuan, & Ming-Syan, 2010) se propone un esquema adaptativo para la extracción de características considerando la distribución de colores en una imagen digital. Los métodos de extracción propuestos, Fixed Cardinality (FC) y Variable Cardinality (VC), se basan en una técnica conocida como binary quaternion-moment-preserving (BQMP) Thresholding, la cual consiste en representar las características de color como cuaterniones, una extensión de los números reales, y representar los parámetros estadísticos de las características de color a través de la definición de los momentos de los cuaterniones con el fin de aplicar el álgebra de los cuaterniones en la resolución de las fórmulas analíticas del BQMP (Soo-Chang & Ching-Min, 1999). Ambos métodos, FC y VC, son capaces de extraer las características de color preservando la distribución de color de una imagen hasta su tercer momento, de tal manera que la distorsión producida por la extracción se ve reducida. Estos dos métodos utilizan la técnica BQMP para extraer las características de color dividiendo de manera iterativa un conjunto de datos en pequeños subconjuntos (proceso adaptativo). Cada método cuenta con distintas condiciones de parada, en el primer caso, el método FC, termina su ejecución cuando un número predefinido de grupos de pixeles (pixel cluster) ha sido extraído. Por otro lado, el método VC, completa su proceso de extracción cuando el número de grupos de pixeles extraído es suficiente para representar la imagen, dicho valor se da cuando la diferencia entre los grupos de pixeles es menor a un valor umbral definido. Los resultados de la ejecución de ambos métodos son dos histogramas de diferentes conjuntos de colores. Finalmente, los autores desarrollan una aplicación de consulta de imágenes basada en ejemplo (CBIR, por sus siglas en inglés) para probar la efectividad de los métodos propuestos, concluyendo que estos mejoraron en un 25% la precisión promedio de recuperación de una imagen sobre cualquier otro sistema tradicional de extracción de características de color.

En (Zhao, Mingbo, Bing, & Peng, 2013) se propone un efectivo mecanismo para la extracción directa de características de imágenes a color basado en la técnica de Análisis de Componentes Principales (PCA , por sus siglas en inglés) ,al cual los autores nombran como ColorPCA. Los autores ponen a prueba el algoritmo desarrollado utilizándolo en la reconstrucción y reconocimiento de imágenes, obteniendo como resultado que el ColorPCA supera en promedio a los métodos tradicionales de PCA y 2DPCA.

Una aplicación del Análisis de Componentes Principales (PCA), se encuentra en (Belkacem-Boussaid, Sertel, Lozanski, Shana'aah, & Gurcan, 2009) , una investigación en la cual se propone un método para la extracción de características de color en texturas de células centroblastos y no centroblastos, para la detección de linfoma folicular. El método se basa en la introducción del análisis por componentes principales en el dominio espectral y tiene por objetivo obtener características de tres espacios de colores diferentes, el RGB, HSI y L*a*b. Las características de los tres espacios son utilizadas luego en su clasificador para detectar los dos tipos de células. Los resultados obtenidos en la clasificación tuvieron una precisión del 82.56%.

El trabajo realizado en (Kobayashi & Otsu, 2009) propone un método para extraer características de color llamado Color Index Local Auto-Correlations (CILAC), el cual varia la forma de utilización de los espacios de color y la indexación de los colores. Este método consta de tres pasos. En primer lugar, los valores de los pixeles son mapeados en espacios de colores como HSV o L*a*b. En segundo lugar, los valores de los pixeles en el espacio de color son indexados en colores básicos tal cual lo hacen los métodos de histogramas, estos valores son descritos en un vector disperso. Por último, se calcula la auto correlación entre los pares de vectores de vecindarios locales. El método propuesto se utilizó para clasificar y recuperar distintos tipos de imágenes como fotografías y pinturas, demostrando en la experimentación que el método CILAC obtiene mejores resultados que los métodos tradicionales.

Un enfoque diferente en cuanto a la extracción de características de color puede encontrarse en (Shih & Chengjun, 2005) . Este trabajo presenta un innovador marco evolutivo para la extracción de características de color, aplicado al reconocimiento de rostros. En primer lugar, se definen dos nuevos espacios de colores como transformaciones lineales a partir del espacio RGB. El primer espacio de color es definido por un canal de luminancia (L) y dos canales de crominancia (C_1,C_2), información del

color. El segundo espacio de color define un canal de luminancia (L) y tres canales de crominancia (C_1,C_2, C_3). Luego, se aplica un algoritmo genético que se encarga de buscar la transformación más óptima del espacio RGB a los dos espacios nuevos. El algoritmo es dirigido por una función objetivo que evalúa la precisión del reconocimiento de un rostro.

3.2 Avances en la caracterización de regiones en base a su morfología

En (Ortiz Zamora, 2002) se describe de manera detallada el fundamento teórico de la Morfología Matemática. Dado que los pixeles son la estructura fundamental de la morfología, se ha destacado la importancia de la noción de orden entre ellos, pues el entretejido interno de una imagen requiere de una ordenación interna de sus componentes. En este trabajo también se presenta la extensión de la Morfología Matemática aplicada a imágenes en color utilizando para ello la información cromática de la familia de modelos HSI (del inglés Hue, Saturation, Intensity – Matiz, Saturación, Intensidad). Asimismo, se ha explicado las características particulares de espacios cromáticos en el procesamiento de imágenes, tratándose de manera eficaz el problema de la indefinición y orden del matiz. Respecto a las estrategias de ordenación vertical, ha sido abordada por medio de un amplio estudio sobre cada una de ellas, de las cuales se ha concluido que el método lexicográfico interactúa mejor con el espacio HSI. Sobre ello, se han mostrado los conceptos de retículo orientado a la intensidad y al matiz, así como también, las mejores permutaciones de todas las señales del modelo cromático. Finalmente, una vez expuesta la metodología a emplear en las operaciones morfológicas en color, se ha comprobó las propiedades de las operaciones, además que se han presentado los primeros residuos cromáticos y definido los filtros morfológicos vectoriales elementales.

En (Aguilar Carrera, 1995) se presentan definiciones de algunos filtros morfológicos que no han sido desarrollados ni tratados por autores precedentes, lo cual proporciona un complemento a lo desarrollado en la caracterización de regiones en base a formas; por ejemplo el caso de los filtros morfológicos suaves recursivos binarios o los filtros de apertura-cierre morfológicos suaves recursivos y no recursivos, esto es aplicable tanto para imágenes binarias, como en tonos de gris.

3.3 Avances en la detección automática de enfermedades en plantas

En (Al Bashish, Braik, & Bani-Ahmad, 2010) se propone un marco de trabajo para la detección y clasificación de cinco tipos de enfermedades en las plantas. El marco está basado en el procesamiento de imágenes y consta de los siguientes pasos; en el primer paso, los autores aplican un técnica de segmentación utilizando el método K-means, de aprendizaje no supervisado, para encontrar los segmentos correspondientes a la hoja en la imagen. Luego, se realiza la extracción de características basada en propiedades específicas de los pixeles, y se aplican análisis estadísticos para determinar cuáles de las características son las más relevantes. Finalmente, se realiza la clasificación utilizando una red neuronal previamente entrenada. Los autores afirman haber obtenido una precisión en la clasificación del 93%.

Otro trabajo propuesto relacionado a la identificación de enfermedades en plantas se encuentra en (Anthonys & Wickramarachchi, 2009). El objetivo de la investigación fue desarrollar un sistema de reconocimiento de imágenes que pueda reconocer enfermedades en arrozales. El procesamiento inicia con la digitalización de las imágenes a color y luego se aplica un método morfológico para segmentar las imágenes. Acto seguido, se extraen características de textura, forma y color que son utilizadas en un método de función de pertenencia (*membership function*) para discriminar entre tres tipos de enfermedades. Finalmente, los autores afirman haber conseguido una precisión del 70% con cerca de cincuenta imágenes de entrenamiento.

En (Haiguang, Guanlin, Zhanhong, & Xiaolong, 2012) se realiza una investigación que tiene por objetivo utilizar el análisis de componentes principales (PCA , por sus siglas en ingles) y redes neuronales para la detección de enfermedades en plantas. Un algoritmo PCA se utiliza para reducir la dimensión de los datos extraídos, en este caso, características de forma, textura y color. El estudio se realizó a partir de la extracción de 50 características; 21 de color, 4 de forma y 25 de textura. Dichas características luego son reducidas con la aplicación del algoritmo PCA y son clasificadas a través de tres tipos de redes neuronales; backpropagation (BP) networks, radial basis function (RBF) neural networks, generalized regression networks(GRNNs) y probabilistic neural networks. En todos los clasificadores utilizados, los autores afirman haber obtenido resultados de precisión mayores al 90% y, en más de un método, del 100%.

En (Golzarian, 2011) se desarrolla un robusto método de segmentación que permite identificar regiones que corresponden a una planta de las regiones que no. El método propuesto es conocido como algoritmo de mapas auto organizado (SOM, por su nombre en inglés), el cual es una estructura especial de red neuronal pero de aprendizaje no supervisado. Las características utilizadas para el estudio fueron todas de color, tomadas de los espacios RGB, HSI y L*a*b. Los autores concluyen que el método usado es sumamente preciso cuando se trata de segmentar dos regiones bien definidas; sin embargo, en el caso de imágenes más complejas, con varias regiones de distintos tamaños, el algoritmo tiene una precisión muy similar a otros algoritmos menos complejos y más comunes.

El estudio realizado en (Rong, Kaneko, Tanaka, Kayamori, & Shimizu, 2013) tiene por objetivo detectar y cuantificar el avance de la Cercospora en imágenes tomadas durante un periodo de diez días a cultivos azucareros. Para ello, plantean la utilización de algoritmos híbridos de reconocimiento por comparación de plantillas y máquinas de vectores de soporte. La investigación se divide en tres partes principales. En la primera parte, se aplica un método de segmentación para distinguir las hojas del fondo de la imagen y obtener una sub-plantilla inicial. En la segunda parte, se aplica el método OCM (del inglés Orientation Code Matching) para realizar el seguimiento del desarrollo de la enfermedad en las sub siguientes imágenes a partir de la plantilla inicial. Finalmente, se aplica un clasificador SVM basado solamente en características de color.

En (Arivazhagan, Shebiah, Ananthi, & Varthini, 2013) se propone una solución de software para detectar y clasificar enfermedades de plantas a partir de un esquema de procesamiento que consiste en cuatro pasos principales. En primer lugar, se crea una transformación estructural para la imagen RGB, luego los pixeles de color verde son enmascarados y removidos utilizando un Thresholding seguido de un proceso de segmentación, luego, el cálculo de las características de textura es realizado para los segmentos relevantes y finalmente las características extraídas son pasadas a través de un clasificador de máquinas de vectores de soporte (SVM, por su nombre en inglés). Los resultados obtenidos en la investigación fueron de un 87%, sobre una base de datos de 500 imágenes de enfermedades en plátanos, frijoles, limones, mangos, papas, entre otras especies

En (Guzmán P, 2003) se propone medir la severidad de la mancha de hierro del cafeto, mediante la utilización del programa Matlab (versión 5. 3), el cual incluye técnicas de

procesamiento de imágenes y reconocimiento por color. La utilización del mencionado programa facilita conocer con mayor certeza y eficiencia el área foliar de las plantas de café y el área afectada por ésta enfermedad. Los resultados de la aplicación de este programa, permite obtener resultados más confiables que pueden aplicarse en trabajos semejantes.

Finalmente, el estudio más cercano a la solución planteada para la problemática, es el de (Hitimana & Gwun, 2014). En este proponen un método automático para la detección del daño ocasionado por la Roya en hojas de cafeto, además, plantean realizar una estimación del grado de severidad. En primer lugar, realizan una mejora de las imágenes de las hojas aplicando un método de mejora de contraste, luego, separan el fondo para quedarse con la región correspondiente a la hoja y finalmente se realiza la detección del área dañada y se estima el grado de severidad. Para detectar el área dañada utilizan el espacio de color YUV, un intermedio entre el modelo RGB y el HSI, en combinación con una técnica de aprendizaje no supervisado conocida como algoritmo Fuzzy C-Mean. Para estimar el grado de severidad, los autores, plantean utilizar un estimador basado en el porcentaje de pixeles correspondientes a una región infectada con respecto a los pixeles correspondientes a la parte sana de la hoja.

3.4 Conclusiones

La revisión de los diversos trabajos y aportes de autores en el mundo, referentes a la resolución de problemas similares, permite tener una base de conceptos y herramientas de partida sobre la cual se orientará el desarrollo del proyecto.

Por esta razón, el estado del arte fue divido en tres partes; la primera orientada a revisar los métodos aplicados en la caracterización de regiones por color, la segunda orientada a revisar los métodos aplicados en la caracterización de regiones según su morfología y, finalmente, la tercera parte es una revisión general de los avances y estudios de métodos para la detección automática de enfermedades en plantas.

4 ESTRUCTURA DEL MODELO ALGORÍTMICO

A continuación se describe el proceso seguido en la evaluación y parametrización de métodos para la elaboración del modelo algorítmico final.

4.1 Principales módulos

El modelo algorítmico desarrollado estará compuesto por una serie de métodos y funciones adecuadamente parametrizadas en base a la experimentación realizada de tal forma que solucionen el problema planteado en la presente investigación. En la Figura 4.1 se observa en resumen los tres grandes módulos que forman el modelo algorítmico desarrollado.

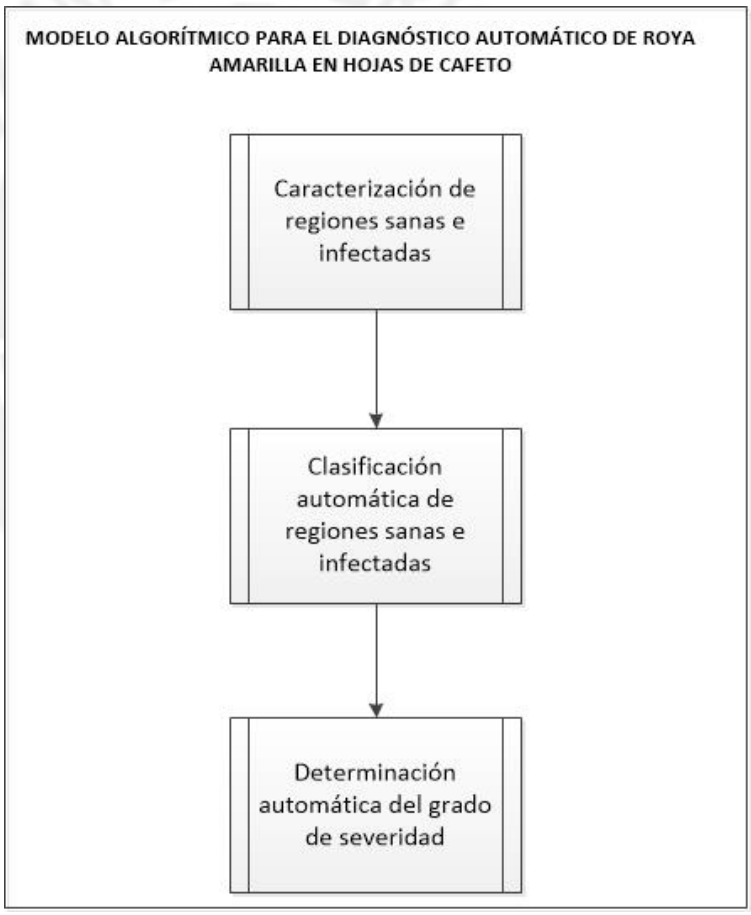

Figura 4.1 Estructura general del modelo algorítmico

4.2 Detalles de los módulos del modelo algorítmico

4.2.1 Caracterización de regiones sanas e infectadas

Este módulo responde a dos objetivos, que son el de conformar el banco de imágenes y el de implementar los descriptores necesarios para extraer características de estas. En la Figura 4.2 se observa el proceso general seguido.

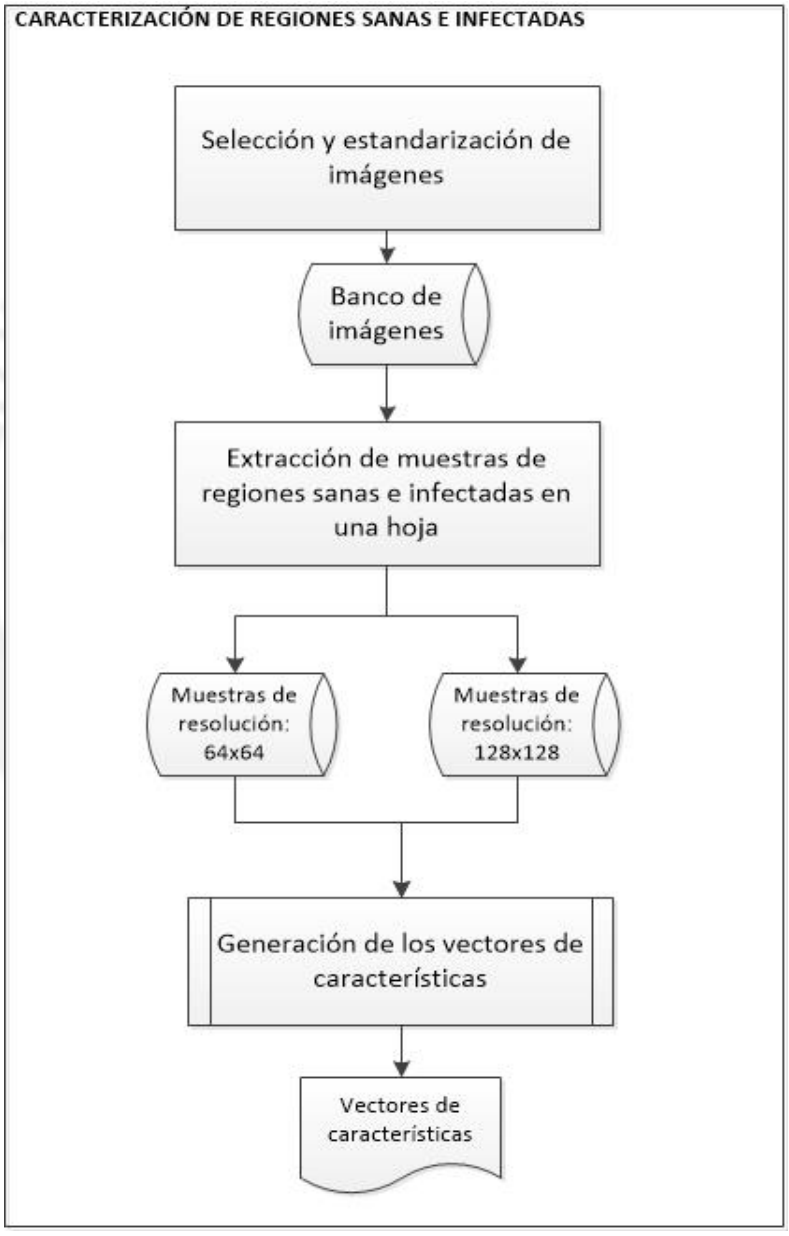

Figura 4.2 Flujo de la caracterización de regiones sanas e infectadas

En la Figura 4.3 se observa a detalle el proceso de generación de los vectores de características, para el cual se implementaron dos descriptores diferentes a través de la combinación de métodos encontrados en la literatura, uno basado en textura y otro basado únicamente en el color de la imagen.

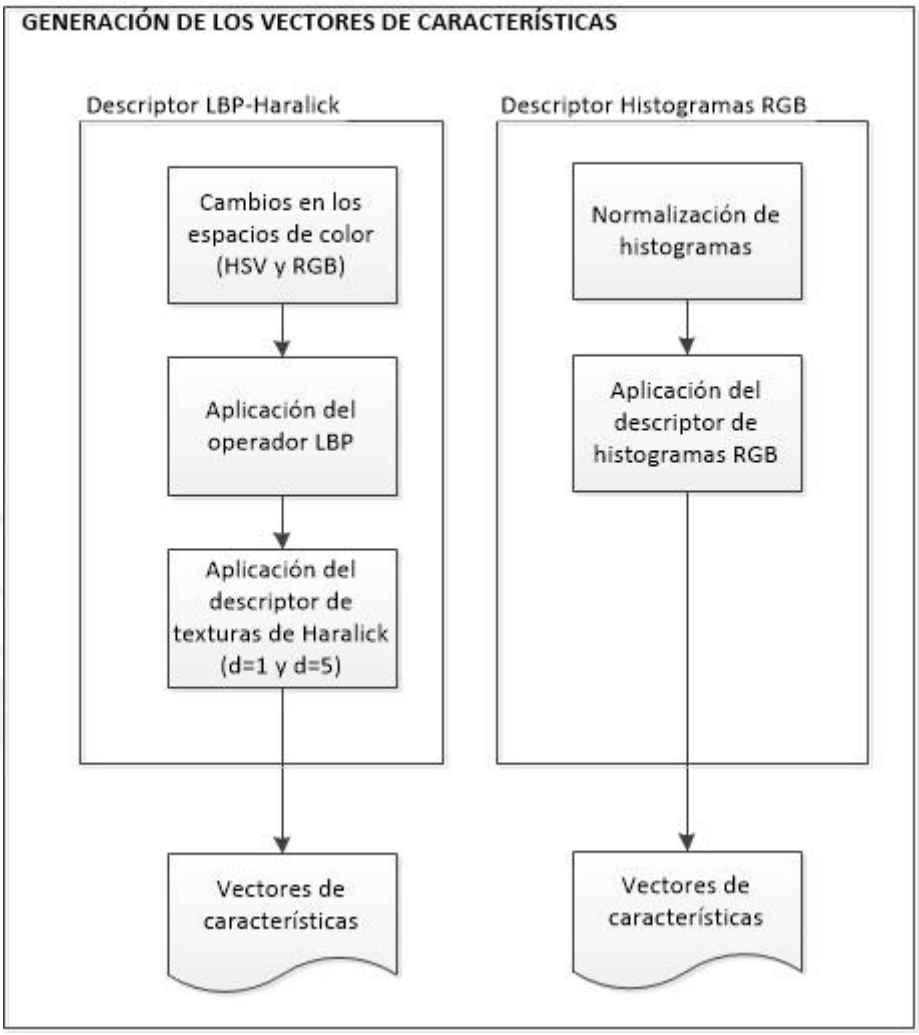

Figura 4.3 Flujo de la generación de vectores de características

4.2.2 Clasificación automática de regiones sanas e infectadas

En la Figura 4.4 se muestra el proceso seguido en la construcción de los diferentes modelos de clasificación, construidos a partir del algoritmo de aprendizaje de máquina conocido como SVM.

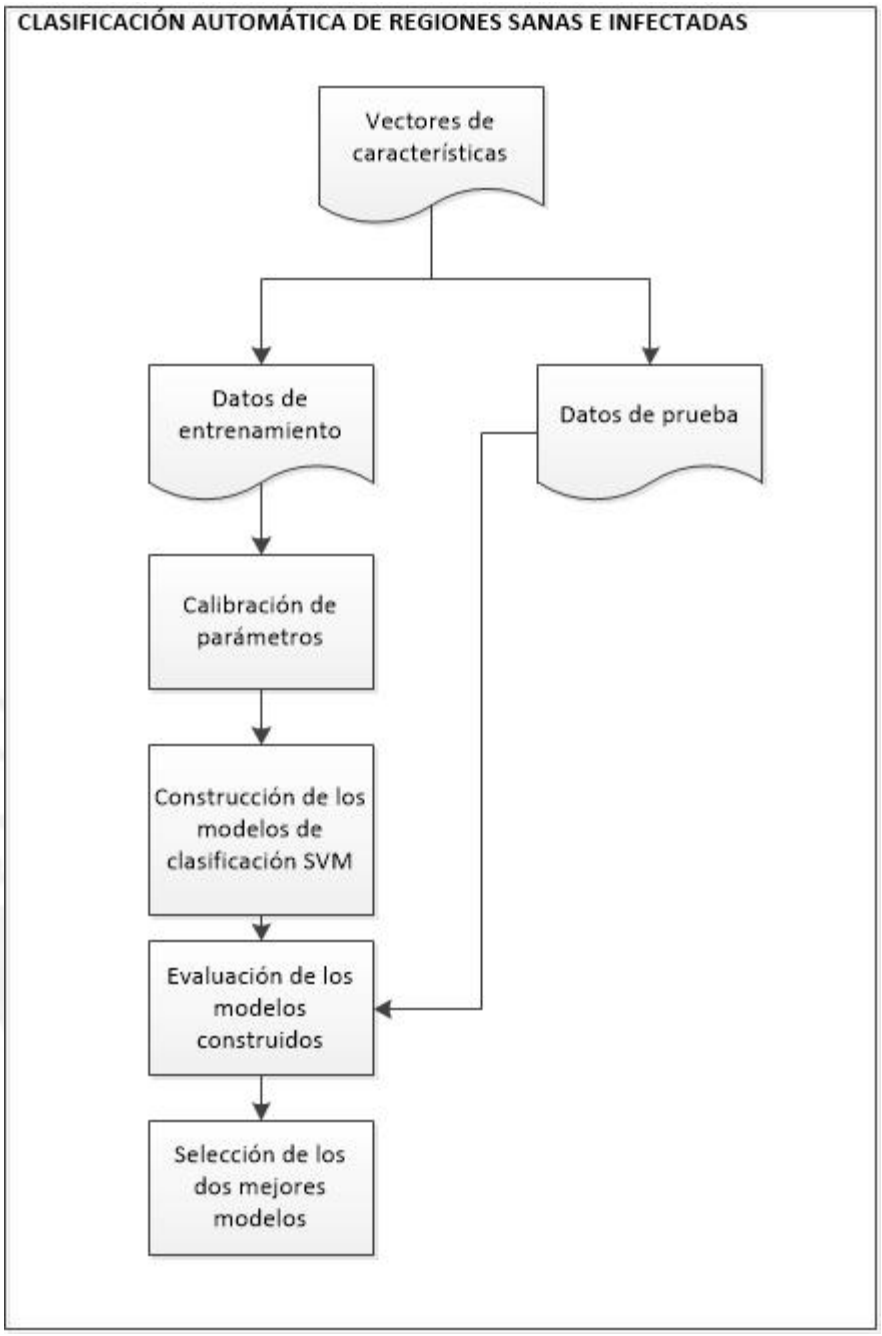

Figura 4.4 Flujo de construcción de los modelos de clasificación

4.2.3 Determinación automática del grado de severidad

En la Figura 4.5 se resume el proceso seguido para la determinación automática del grado de severidad. Este módulo responde a los últimos objetivos del proyecto, los

cuales son aplicar dos métodos de segmentación y cuantificar el grado de severidad en una hoja infectada.

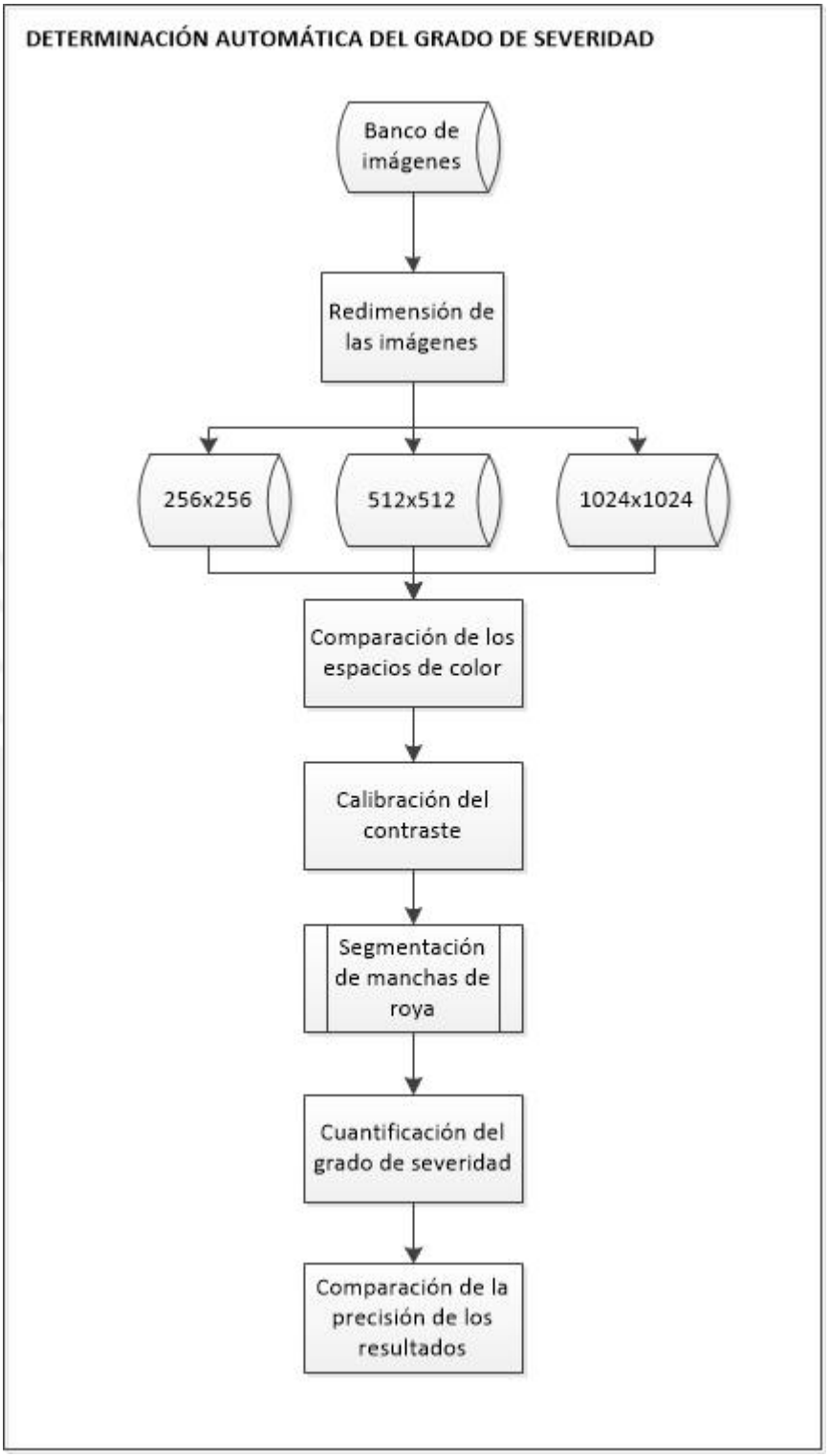

Figura 4.5 Flujo seguido para la determinación del grado de severidad

En la Figura 4.6 se muestra a detalle el proceso de segmentación de manchas de Roya Amarilla.

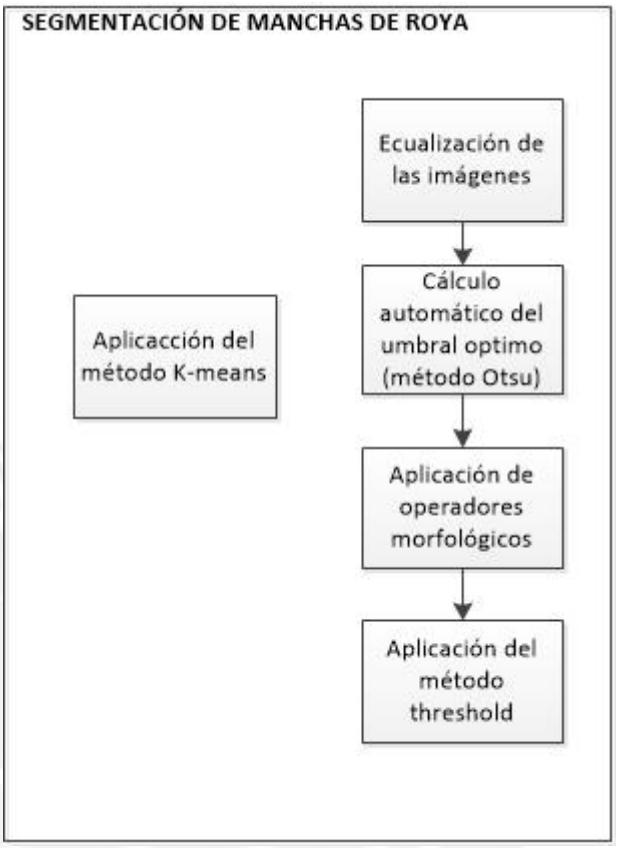

Figura 4.6 Flujo de métodos aplicados en la segmentación de manchas de Roya.

5 CARATERIZACIÓN DE REGIONES SANAS E INFECTADAS

El presente capítulo se divide en tres secciones, las cuales serán desarrolladas a continuación.

5.1 Selección y estandarización del conjunto de imágenes

A continuación se describen las características de los conjuntos de imágenes que fueron utilizados en la investigación.

5.1.1 Sobre el conjunto de imágenes a utilizarse

En la Figura 5.1 se muestran las imágenes que fueron tomadas en un ambiente no controlado con cámaras de 5, 8 y 12 MP (Mega-Pixeles). En total se consiguió tomar 250 fotografías.

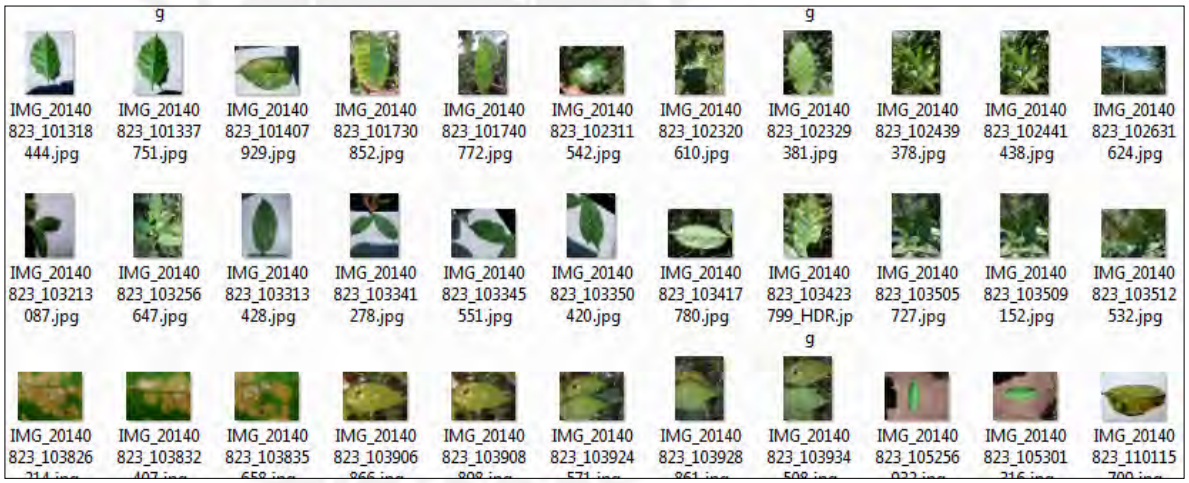

Figura 5.1 Fotografías tomadas en Chanchamayo, Perú (2014).

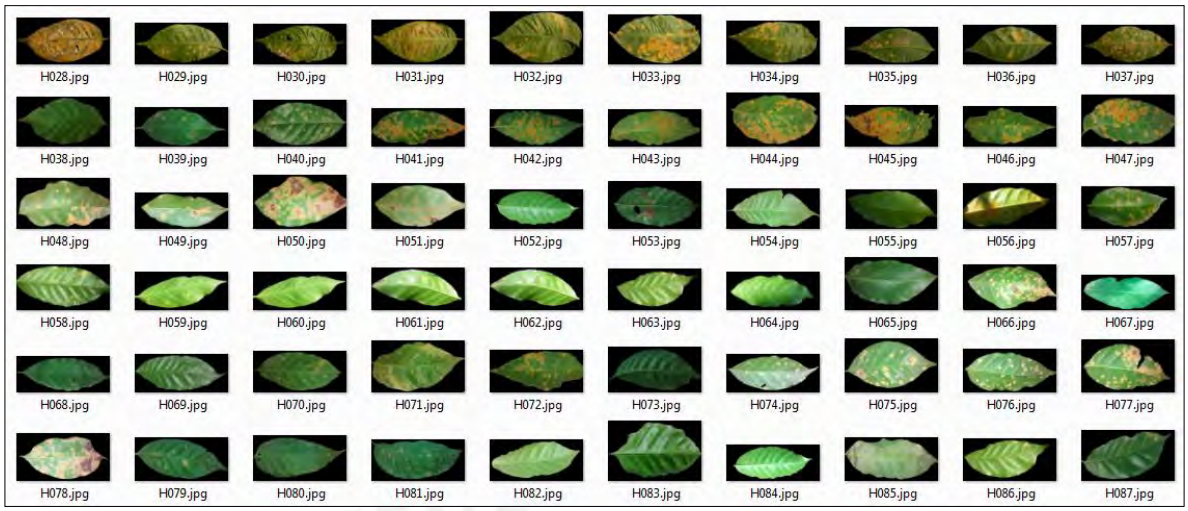

Figura 5.2 Fotografías Procesadas con fondo negro.

Dado que el alcance del proyecto no contemplaba trabajar sobre imágenes tomadas en ambientes no controlados, se procedió a simular, a través de la selección y edición manual de las imágenes, un ambiente al cual se le podría catalogar como semi-controlado. Primero, se seleccionaron las imágenes en las cuales la iluminación no afectaba en sobremanera la visualización de la hoja o de las manchas de Roya. Luego, se procedió a remover el fondo de la fotografía, quedando únicamente la hoja como objeto de interés. Finalmente, el corpus de imágenes se vio reducido a un total de 90 fotos. En la Figura 5.2 se puede observar el corpus de imágenes final.

5.1.2 Sobre la resolución y estandarización de las imágenes

Para poder tener mayor variedad en las pruebas experimentales, se procedió a redimensionar las imágenes y estandarizarlas en tres grupos de diferentes resoluciones. El primer grupo de 256 pixeles de ancho y el segundo de 512, y finalmente el último de 1024, cada uno compuesto por 90 imágenes.

Para automatizar este proceso, se desarrolló un script en Python, haciendo uso de la librería OpenCV, que permite redimensionar las imágenes y mantener la relación de aspecto.

5.1.3 Sobre las muestras extraídas para la etapa de clasificación

Del conjunto de imágenes, se tomaron muestras de diferentes regiones de las hojas, a las cuales se llamó y dividió en dos, regiones sanas, o libres de Roya, y regiones infectadas. La razón de este muestreo por regiones es porque en la etapa de

clasificación se distinguirá si una región está infectada o no y luego se procederá a dar el diagnóstico de toda la hoja, de este modo se puede contar con un mayor número de imágenes para la parte de entrenamiento y posteriores pruebas en los clasificadores.

Se determinó formar dos conjuntos de muestras, el primero de una resolución de 64x64 y el segundo de 128x128. Cada conjunto está formado por el mismo número de regiones infectadas y sanas. En la tabla 5.1 se resume lo dicho líneas arriba

Tabla 5.1 Número de muestras por Región para cada resolución

	Nro. Muestras por región		
Resolución	**Infectada**	**Sana**	**Total**
64x64	3200	3200	6400
128x128	1500	1500	3000

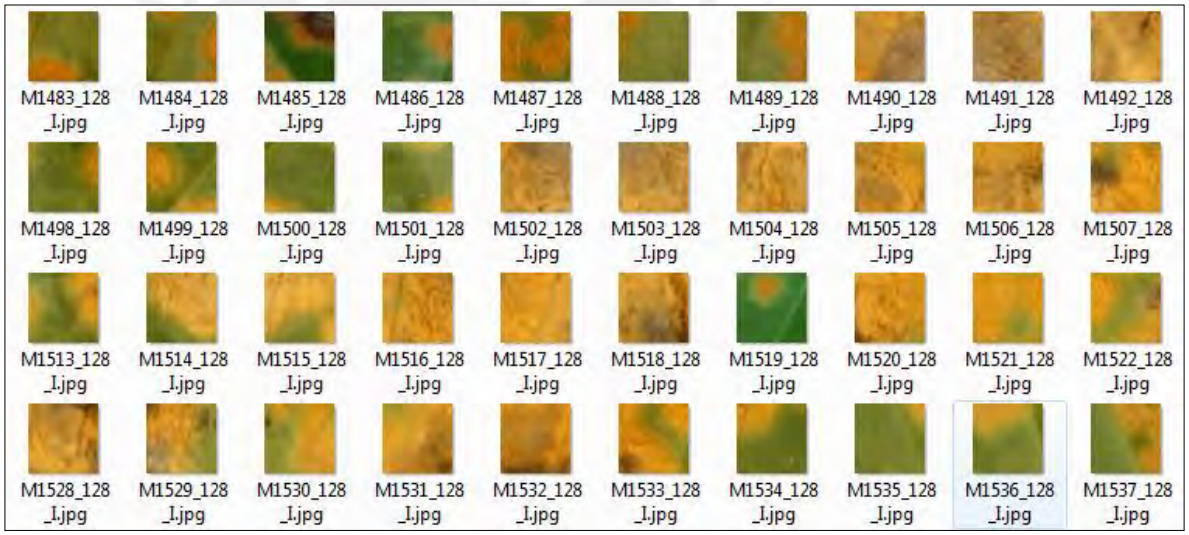

Figura 5.3 Regiones de Hoja infectadas con Roya

Para realizar este muestreo en las imágenes, se desarrolló un script en Python el cual se encarga de automatizar el proceso de extracción de muestras, ya que recorre toda la imagen y la subdivide en las regiones de los tamaños deseados. En la Figura 5.3 se observa el conjunto de muestras obtenido luego de aplicar el script sobre una hoja infectada con Roya.

Luego de obtener las muestras con el script, se procedió a separar las muestras sanas de las infectadas en distintas carpetas.

5.2 Extracción de características mediante el uso del descriptor LBP-Haralick

Según (Haralick, Shanmugam, & Dinstein, 1973) la textura y los niveles de gris (o tonos de gris) en una imagen están altamente relacionados. En el caso de una región infectada con Roya Amarilla, una característica resaltante, además de su color, es la textura rugosa del hongo, que difiere con respecto a una región sana. Por este motivo, se optó por utilizar el descriptor de textura de Haralick como extractor de características. Por otro lado, para agudizar las diferencias de textura, se pensó en aplicar el operador de Patrones Binarios Locales LBP (del inglés *Local Binary Pattern*) como paso previo al uso de Haralick. Esta idea se vio sustentada en la investigación de (Porebski et al., 2008). En ella hacen uso de un operador LBP modificado para tomar en cuenta, además de las características de textura, variaciones locales de color, con el fin de clasificar imágenes con diferentes texturas y colores.

Tanto la variante del operador LBP como el descriptor de Haralick fueron implementados haciendo uso de las herramientas descritas en el primer capítulo. A continuación, se describe todo el proceso, desde la implementación de los métodos hasta la extracción y generación de los vectores de características.

5.2.1 Implementación del operador LBP

Se implementó la variación del operador LBP propuesta en (Porebski et al., 2008) que permite tomar en cuenta variaciones locales de color.

Para implementar este operador se utilizaron tres estructuras. La primera, una matriz a manera de máscara de pesos, que van del 1 al 128. La segunda estructura es igualmente una matriz, en la que se almacenan los valores obtenidos a partir de la umbralización. Finalmente, la tercera estructura almacena los valores finales, obtenidos de la codificación LBP.

Las estructuras se muestran en las Figuras 5.4, 5.5 y 5.6 que se muestra a continuación:

- weightMask[$]_{3x3}$: Matriz de pesos

1	2	4
8		16
32	64	128

Figura 5.4 weightMask[$]_{3x3}$

- lbpMatrix[$]_{3x3}$: Matriz que almacena los valores obtenidos luego de la umbralizacion

0	1	0
0		1
0	0	0

Figura 5.5 lbpMatrix[$]_{3x3}$

- imageLBP[$]_{mxn}$: Matriz en la cual se almacena la intensidad de los pixeles luego de aplicar el operador sobre la imagen

	P1	P2	P3	
	P4	P5	P6	
	P7	P8	P9	

Figura 5.6 imageLBP[$]_{3x3}$

Dónde mxn representa la resolución de la imagen original, y Pi representa el valor del pixel luego de aplicarle el operador LBP.

La generación de las imágenes LBP usando el operador modificado para tomar en cuenta variaciones de color se basa en los siguientes pasos (Porebski et al., 2008):

- Leer la imagen de entrada y recorrer cada uno de los pixeles P.

- Por cada pixel P, se compara la magnitud de su vector de color C(P) = [C1(P),C2(P),C3(P)] con la magnitud del vector de color de cada uno de los pixeles vecinos C(P').
- Se aplica la relación de orden: Si Magnitud(C(P))<=Magnitud(C(P')) entonces C(P')= 1 caso contrario C(P')= 0.
- El resultado de cada umbralización es codificado gracias a la máscara de pesos.
- Los valores ponderados, gracias a la máscara de pesos, son almacenados en la estructura final.

El resultado de este proceso es una imagen texturizada. En la Figura 5.7 se ve el resultado obtenido, a la izquierda la imagen original y a la derecha la imagen texturizada por el operador LBP.

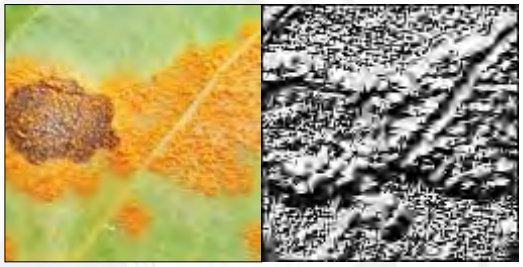

Figura 5.7 Imagen Original y Texturizada

5.2.2 Implementación del descriptor de textura de Haralick

Según lo explicado en el capítulo 2, las matrices GLCM (del inglés *Gray Level Co-Ocurrence Matrix*) permiten cuantificar la frecuencia con la cual diferentes combinaciones de niveles de gris ocurren en una imagen (Haralick et al., 1973). En el presente trabajo, las imágenes fueron procesadas teniendo en cuenta 256 niveles de gris, que van del 0 al 255, por este motivo se definió una estructura de datos de la siguiente forma:

- Matriz GLCMfreq $[x, y]_{256x256}$: Matriz que almacena el número de ocurrencias de un pixel de referencia con un pixel vecino.

Tabla 5.2 Matriz de Ocurrencia del Par de Píxeles (x, y)

	Nivel 0	Nivel 1	…	Nivel 255
Nivel 0	$F_\theta^d(0,0)$	$F_\theta^d(0,1)$		$F_\theta^d(0,255)$
Nivel 1		$F_\theta^d(0,0)$		
…				
Nivel 255				$F_\theta^d(255,255)$

En la Tabla 5.2 se tiene $F_\theta^d(x,y)$= Frecuencia de ocurrencia del par de pixeles (x,y) en una dirección ɵ con una separación *d.*

La implementación del método recibe como parámetros la dirección (ɵ) y la separación entre los pixeles (*d*); sin embargo, solo toma en cuenta cuatro direcciones que son θ=0°, θ=45°, θ=90° y θ=135°.

La siguiente estructura necesaria es la matriz de frecuencias relativas, sobre la cual se realizan todos los cálculos para extraer las características de textura. Dicha matriz toma la siguiente forma:

- Matriz GLCMpij$[x,y]_{256x256}$: Matriz que almacena la frecuencia relativa o propabilidad de courrencia de un pixel de referencia con un pixel vecino.

Tabla 5.3Matriz de Ocurrencia de Frecuencia Relativa

	Nivel 0	Nivel 1	…	Nivel 255
Nivel 0	$P_\theta^d(0,0)$	$P_\theta^d(0,1)$		$P_\theta^d(0,255)$
Nivel 1		$P_\theta^d(0,0)$		
…				
Nivel 255				$P_\theta^d(255,255)$

En la Tabla 5.3 se tienen $P_\theta^d(x,y)$=$F_\theta^d(x,y)$/ S = Frecuencia relativa o probabilidad de ocurrencia del par de pixeles (x,y) en una dirección ɵ con una separación *d. Donde S = Suma total de ocurrencias*

Luego de generadas las matrices para un determinado "θ" y una determinada separación "*d*", se procedió a la extracción de características o valores estadísticos que mejor representen la imagen. Las características extraídas por cada matriz generada fueron las siguientes:

- Contraste:

$$f_1 = \sum_{n=0}^{N_{g-1}} n^2 \{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \}$$

Dónde: $|i - j| = n$

- Disimilaridad:

$$f_2 = \sum_{n=0}^{N_{g-1}} n \{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \}$$

Dónde: $|i - j| = n$

- Homogeneidad:

$$f_3 = \sum_i \sum_j \frac{1}{1 + (i-j)^2} p(i,j)$$

- Segundo momento angular:

$$f_4 = \sum_i \sum_j \{p(i,j)\}^2$$

- Energía

$$f_5 = \sqrt{\sum_i \sum_j \{p(i,j)\}^2}$$

- Máxima probabilidad

$$f_6 = Max\_prob(p(i,j))$$

- Entropía:

$$f_7 = -\sum_i \sum_j p(i,j) log\big(p(i,j)\big)$$

- Suma promedio:

$$f_8 = \sum_{i=2}^{2N_g} ip_{x+y}(i)$$

- Suma de cuadrados: Varianza

$$f_9 = \sum_i \sum_j (i-u)^2 p(i,j)$$

- Desviación estándar:

$$f_{10} = \sqrt{\sum_i \sum_j (i-u)^2 p(i,j)}$$

- Correlación:

$$f_{11} = \frac{\sum_i \sum_j (ij) p(i,j) - u_x u_y}{\sigma_x \sigma_y}$$

5.2.3 Generación de los vectores de características

Con respecto a la generación de los vectores de características, se tomaron en cuenta dos propiedades:

- El espacio de color sobre el cual se extraerían las características.
- La distancia *d* entre el pixel de referencia y los pixeles vecinos para el cálculo de las matrices de co-ocurrencia.

Los espacio de color elegidos fueron el RGB (del inglés *Red*, *Green* y *Blue*) y el HSV (del inglés *Hue*, *Saturation*, *Value*). Parte de la investigación buscó determinar con cuál de los dos espacios de color se obtienen mejores resultados al momento de discriminar una región con Roya Amarilla de una sana.

Por otro lado, se utilizaron dos distancias de separación entre pixel vecino y de referencia (d=1 y d=5) para calcular las matrices de co-ocurrencia. En (Porebski et al., 2008) comparan cinco distancias (d=1,2,3,4,5), obteniendo los mejores resultados para d=1 y d=5 en un problema de clasificación multi-clase. Por ello, se decidió reducir el número de distancias a probar a solo dos, que son las que dieron mejor resultado en la investigación citada líneas arriba.

Finalmente, lo que se obtiene son cuatro vectores (dos para cada espacio de color) de treinta y tres características cada uno. Los pasos seguidos fueron los siguientes:

- Leer conjunto de imágenes
- Por cada imagen:
    - Crear una copia en el espacio de color HSV
    - Por cada espacio de color (HSV y RGB):
        - Aplicar el operador LBP y obtener imagen texturizada
        - Por cada valor de separación "d" (1 o 5):
            - Extraer características con el descriptor de Haralick.
            - Por cada característica extraída de las cuatro matrices de co-ocurrencia (una por cada dirección "θ") calcular la media, rango y desviación estándar.
            - Almacenar los valores obtenidos en el vector de características
            - Imprimir los valores en un archivo csv

Este proceso se aplicó a cada conjunto de muestras a través de un script desarrollado en Python, el cual finalmente almacena cada vector de características en un archivo csv.

5.3 Extracción de características mediante el uso de histogramas RGB

Con el fin de comparar la efectividad del descriptor de texturas de Haralick aplicado sobre imágenes LBP, se decidió formar otro descriptor más sencillo basado únicamente en los

histogramas de color RGB de las muestras. A continuación se detallan los pasos seguidos para crear el vector de características.

5.3.1 Análisis y comparación de los histogramas por canal

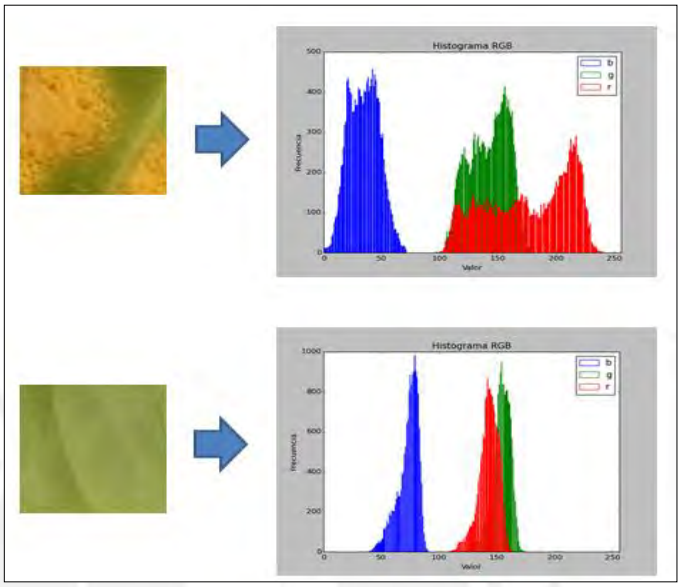

Figura 5.8 Histogramas para cada canal de color (R, G, B)

En la Figura 5.8 se observan los histogramas para cada canal de color (R, G, B), tanto en las regiones sanas como en regiones infectadas por Roya Amarilla.

A pesar de que el espacio de color RGB es demasiado sensible a cambios en la iluminación, y los histogramas pueden variar mucho de una imagen a otra, según (Finlayson, Schiele, & Crowley, 1998) este efecto se puede reducir normalizando la imagen, de tal manera que se alcancen mejores resultados al momento de discriminar entre la región sana de la infectada. Por ello, antes de comparar los histogramas de ambas regiones, se procedió a normalizarlos, técnica que consiste en extender el valor de la intensidad de los pixeles por cada canal para cubrir todo el rango de valores de pixeles. En la Figura 4.10 se observa el histograma normalizado de la región infectada y en la Figura 4.11 se observa el histograma normalizado de la región sana.

Por simple inspección se observa que en las regiones infectadas los valores de los pixeles se encuentran más dispersos, se reparten por todos los 255 niveles; mientras que, en la región sana, los valores se encuentran más concentrados en ciertos rangos. Además, se observa que en el canal R la intensidad de una región infectada es mayor a

la de una región sana. En el caso del canal B, la intensidad de los pixeles es menor en la región infectada. En cuanto al canal G no se observa mayor diferencia en el rango de valores para ambas regiones. Estas características en la intensidad de los pixeles por cada canal brindan una idea de que tan bien podría discriminar un clasificador entre ambas regiones utilizando solo estos valores.

5.3.2 Generación de los vectores de características

Por lo discutido en la sub-sección anterior, se decidió formar un vector de características compuesto por el promedio de la intensidad de los pixeles en cada canal y por sus respectivas desviaciones estándar. Los atributos o características del vector quedan resumidos de la siguiente forma:

- R_mean = $\sum_{i=0}^{i=n-1} p_i[R]/n$
- G_mean = $\sum_{i=0}^{i=n-1} p_i[G]/n$
- B_mean = $\sum_{i=0}^{i=n-1} p_i[B]/n$
- R_stddev = $\sum_{0}^{n-1} (p_i[R] - R_{mean})^2/n$
- G_stddev = $\sum_{0}^{n-1} (p_i[G] - G_{mean})^2/n$
- B_stddev = $\sum_{0}^{n-1} (p_i[B] - B_{mean})^2/n$

Donde:

- $p_i[C]$ es el valor de la intensidad del pixel *i* en el canal C.
- *n* es el número total de pixeles en la imagen

Para automatizar este proceso, se desarrolló un script en Python el cual normaliza cada imagen del conjunto de muestras y luego calcula los valores correspondientes al vector de características. Finalmente estos valores son almacenados en un archivo de formato separado por comas (.csv).

6 CLASIFICACIÓN AUTOMÁTICA DE REGIONES INFECTADAS Y SANAS

El presente capitulo consta de tres secciones en las cuales se desarrolla todo el proceso de aprendizaje de máquina.

6.1 Construcción de los modelos de clasificación

Para la construcción de los modelos de clasificación se optó por el algoritmo de aprendizaje de máquina conocido como SVM (del inglés *Support Vector Machines*), ya que, según la bibliografía, es uno de los más populares en la actualidad puesto que ofrece un mejor rendimiento para conjuntos de datos de mediano tamaño en comparación con la mayoría de algoritmos (Marsland, 2009).

6.1.1 Selección del kernel

El kernel seleccionado para este trabajo fue el RBF (del inglés *Radial Basis Function*), ya que según (Hsu et al., 2003) ofrece las siguientes características:

- Mapea las muestras a un espacio dimensional más alto, por lo cual puede manejar el caso en el cual la relación entre los atributos y la clase no es lineal.
- Es una forma generalizada del kernel lineal, ya que este se puede comportar como un kernel lineal para ciertos parámetros γ y C.
- El número de hiper-parámetros no es igual de alto que en el caso del kernel polinomial, lo cual influencia para bien en la complejidad del modelo.
- Tiene menor complejidad numérica, pues varía entre 0 y 1, a diferencia del kernel polinomial que puede tomar valores infinitos dependiendo del grado.

6.1.2 Conjunto de datos de entrenamiento y de prueba

Se separaron los datos en conjuntos de entrenamiento y de prueba, los primeros utilizados en la construcción de los modelos y los segundos utilizados en la validación y prueba de los modelos. Además, se determinó separar las muestras de datos por la mitad.

En la Tabla 6.1, Tabla 6.2 y la Tabla 6.3 se muestran los conjuntos de datos:

Tabla 6.1 Conjunto de datos Haralick – LBP para muestras de 128x128

	RGB		HSV	
	d=1	d=5	d=1	d=5
Datos de entrenamiento	1500	1500	1500	1500
Datos de prueba	1500	1500	1500	1500
Total	3000	3000	3000	3000

Tabla 6.2 Conjunto de datos Haralick – LBP para muestras de 64x64

	RGB		HSV	
	d=1	d=5	d=1	d=5
Datos de entrenamiento	3200	3200	3200	3200
Datos de prueba	3200	3200	3200	3200
Total	6400	6400	6400	6400

Tabla 6.3 Conjunto de datos RGB Normalizados

Resolución	64x64	128x128
Datos de entrenamiento	3200	1500
Datos de prueba	3200	1500
Total	6400	3000

Por lo tanto, se cuenta con diez conjuntos de datos diferentes, cada uno dividido en datos de entrenamiento y de prueba. Los conjuntos de datos generados a partir del descriptor de Haralick sobre imágenes LBP cuentan con 34 características por instancia, incluyendo la clase (infectada o sana); mientras que, los conjuntos de datos generados a partir del descriptor de histogramas RGB cuentan con 7 características por instancia, incluyendo la clase (infectada o sana).

6.1.3 Calibración de parámetros

Cuando el kernel RBF es utilizado, hay dos parámetros que necesitan ser correctamente seleccionados, estos son el (γ y C) para los cuales se hizo la respectiva calibración dado que no se sabe a priori los mejores valores de γ y C para este problema en particular.

Se optó por aplicar una técnica conocida como “Búsqueda en Grilla”, en la cual para un rango de valores de γ y un rango de valores de C, se construye un modelo para cada una de las combinaciones posibles con el objetivo de seleccionar la mejor. Se utilizó la muestra de entrenamiento para realizar la calibración de los parámetros pero aplicando una validación cruzada estratificada, que según (Kohavi, 1995) en general es un mejor esquema de validación, en términos de sesgo y varianza, comparado con la validación cruzada regular. La estratificación es el proceso de re-ordenar los datos de tal forma que cada sub-grupo (*fold*) sea representativo de toda la muestra.

La herramienta utilizada para este proceso fue la librería Scikit-Learn, la cual brindó las clases y métodos necesarios, estos se muestran a continuación:

class sklearn.cross_validation.StratifiedKFold(*y*, *n_folds=3*)

- y: etiquetas de clases del conjunto de entrenamiento.
- n_folds: número de subgrupos (*folds*) en los cuales es dividida la muestra.

class sklearn.grid_search.GridSearchCV (*estimator=, param_grid=, cv=*)

- estimator: clasificador sobre el cual se probara cada combinación de parámetros a calibrar
- param_grid: diccionario con los nombres y rango de valores de los parámetros a calibrar
- cv: entero u objeto generador de validación. Si es enviado un número, se aplica el método de validación cruzada con ese número de subgrupos.

Siguiendo la recomendación encontrada en (Hsu et al., 2003), el rango de valores evaluados para C fue $[2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{11}, 2^{13}, 2^{15}]$ y el rango de valores evaluados para γ fue $[2^{-15}, 2^{-13}, 2^{-11}, 2^{-9}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}]$. Las matrices o “Grillas de Búsqueda” de calibración para cada uno de los modelos se pueden encontrar en los anexos.

Luego de encontrar los mejores parámetros (γ, C) se utilizó nuevamente todo el conjunto de datos de entrenamiento para construir cada clasificador.

6.1.4 Modelos construidos

Dado que se cuenta con diez conjuntos de datos diferentes, se procedió a la construcción de diez modelos basados en el algoritmo SVM. Los ocho primeros modelos utilizan las características extraídas mediante el descriptor de Haralick sobre imágenes LBP, con las variantes del espacio de color, la resolución de las muestras y la separación entre el pixel de referencia y los pixeles vecinos. Los últimos dos modelos fueron construidos a partir de las características extraídas con del descriptor de histogramas RGB, y varían entre ellos por la resolución de las muestras. En la tabla 6.4 se resume lo explicado líneas arriba.

Tabla 6.4 Resumen de los modelos construidos

Modelo - Clasificador	Descripción
SVM_HLBP_64x64_RGB_d1	Modelo construido con características extraídas por el descriptor Haralick con separación d=1, en imágenes de 64x64 de resolución sobre el espacio RGB.
SVM_HLBP_64x64_RGB_d5	Modelo construido con características extraídas por el descriptor de Haralick con separación d=5, en imágenes de 64x64 de resolución sobre el espacio RGB.
SVM_HLBP_64x64_HSV_d1	Modelo construido con características extraídas por el descriptor Haralick con separación d=1, en imágenes de 64x64 de resolución sobre el espacio HSV.
SVM_HLBP_64x64_HSV_d5	Modelo construido con características extraídas por el descriptor Haralick con separación d=5, en imágenes de 64x64 de resolución sobre el espacio HSV.
SVM_HLBP_128x128_RGB_d1	Modelo construido con características extraídas por el descriptor Haralick con separación d=1, en imágenes de

	128x128 de resolución sobre el espacio RGB.
SVM_HLBP_128x128_RGB_d5	Modelo construido en base al descriptor de Haralick con separación d=5, en imágenes de 128x128 de resolución sobre el espacio RGB.
SVM_HLBP_128x128_HSV_d1	Modelo construido en base al descriptor de Haralick con separación d=5, en imágenes de 128x128 de resolución sobre el espacio HSV.
SVM_HLBP_128x128_HSV_d5	Modelo construido en base al descriptor de Haralick con separación d=5, en imágenes de 128x128 de resolución sobre el espacio HSV.
SVM_HISTRGB_64x64	Modelo construido en base al descriptor de histogramas RGB imágenes de 64x64.
SVM_HISTRGB_128x128	Modelo construido en base al descriptor de histogramas RGB imágenes de128x128.

Todos los modelos generados fueron nombrados a partir de la siguiente nomenclatura:

- SVM_DESC_RES_ESPACIO_dx, en donde SVM es el nombre del algoritmo de aprendizaje de máquina, DESC es el nombre del descriptor que extrajo las características para el modelo, RES es la resolución de las imágenes sobre las que se usó el descriptor, ESPACIO es el espacio de color sobre el que se aplicó el descriptor, y dX es la distancia de separación entre el pixel de referencia y el pixel vecino. Los últimos dos atributos solo aplican en el caso de que el descriptor sea el de Haralick.

La herramienta utilizada para la construcción de los modelos fue la librería Scikit-Learn, la cual brinda clases y funciones especializadas para tareas de aprendizaje de máquina.

La clase brindada por la librería para realizar el aprendizaje de máquina con el algoritmo SVM es la siguiente (Pedregosa et al., 2011):

class sklearn.svm.SVC(*C=1.0*, *kernel='rbf'*, *gamma=0.0*, *shrinking=True*, *tol=0.001*,)

- C : Para metro de penalización asociado al error
- Kernel: Especifica el tipo de kernel a ser usado por el algoritmo.
- Gamma: Parámetro del kernel. Si es cero se calcula como 1/nro_caracteristicas.
- Shrinking: Heurística para acelerar el proceso de optimización.
- Tol: Criterio de tolerancia para terminar el proceso.

Además, se utilizaron los siguientes métodos de esta clase:

- fit(X, y): Realiza el entrenamiento del clasificador a partir de los datos de entrada (X=instancias , y=etiqueta de clase)
- predict(X): Clasifica la instancia X dada como input.
- score(X, y): Devuelve la exactitud (*Accuracy*) promedio obtenida a partir de los datos de entrada.

Un paso previo antes de iniciar la construcción de los clasificadores fue la calibración de los parámetros, proceso que es descrito a continuación.

6.2 Evaluación de los modelos construidos

Tal como se explicó en la sección anterior, los modelos fueron evaluados con un conjunto de datos diferente al conjunto con el que fueron construidos, pero ambos contenían la misma cantidad de datos. Los resultados y reportes obtenidos a partir de la evaluación de cada uno de los modelos se pueden encontrar en los anexos.

La herramienta utilizada para este proceso fue la librería Scikit-Learn, la cual cuenta con un módulo especializado en métricas de clasificación. Las funciones utilizadas de dicho modulo fueron las siguientes:

metrics.accuracy_score(*y_true, y_pred*)

metrics.classification_report(*y_true, y_pred*)

metrics.confusion_matrix(*y_true, y_pred*)

metrics.roc_curve(*y_true, y_score*)

- y_true: valores reales o correctos
- y_pred: valores estimados
- y_score: arreglo de scores objetivo, pueden ser una probabilidad.

6.3 Análisis comparativo de resultados

Luego de obtener los resultados de la evaluación para cada modelo, se procedió a la comparación de los valores obtenidos con el fin de determinar cuál de los modelos construidos tuvo el mejor rendimiento.

6.3.1 Selección del mejor modelo de clasificación

En la Tabla 6.5 y Tabla 6.6 se resumen los resultados obtenidos por cada modelo.

Tabla 6.5 Matriz comparativa de resultados de clasificación I.

Modelo	Accuracy (ACC)	Error Rate (ER)
Muestras de resolución 64x64		
SVM_HLBP_64x64_RGB_d1	74.53%	25.47%
SVM_HLBP_64x64_RGB_d5	76.59%	23.41%
SVM_HLBP_64x64_HSV_d1	76.09%	23.91%
SVM_HLBP_64x64_HSV_d5	80.94%	19.06%
SVM_HISTRGB_64x64	87.50%	12.50%
Muestras de resolución 128x128		
SVM_HLBP_128x128_RGB_d1	53.47%	46.53%
SVM_HLBP_128x128_RGB_d5	73.47%	26.53%
SVM_HLBP_128x128_HSV_d1	74.60%	25.40%
SVM_HLBP_128x128_HSV_d5	89.40%	10.60%
SVM_HISTRGB_128x128	95.53%	4.47%

Tabla 6.6 Matriz comparativa de resultados de clasificación II.

Modelo	Nro. instancias	Precision	Recall	F-measure	AUC
Muestras de resolución 64x64					
SVM_HLBP_64x64_RGB_d1	3200	0.75	0.75	0.75	0.745
SVM_HLBP_64x64_RGB_d5	3200	0.77	0.77	0.77	0.766
SVM_HLBP_64x64_HSV_d1	3200	0.77	0.76	0.76	0.761
SVM_HLBP_64x64_HSV_d5	3200	0.81	0.81	0.81	0.809
SVM_HISTRGB_64x64	3200	0.88	0.88	0.88	0.875
Muestras de resolución 128x128					
SVM_HLBP_128x128_RGB_d1	1500	0.53	0.53	0.53	0.535
SVM_HLBP_128x128_RGB_d5	1500	0.81	0.80	0.80	0.802
SVM_HLBP_128x128_HSV_d1	1500	0.75	0.75	0.75	0.746
SVM_HLBP_128x128_HSV_d5	1500	0.90	0.89	0.89	0.894
SVM_HISTRGB_128x128	1500	0.96	0.96	0.96	0.955

Claramente los modelos con mejores resultados fueron los basados en el descriptor de histogramas RGB, ya que obtuvieron una mayor puntuación en todas las medidas tanto para las muestras de 64x64 como para las muestras de 128x128 de resolución. Sin embargo; una desventaja de este descriptor es que no toma en cuenta las características de textura, con lo cual es probable que cualquier mancha amarilla o traslucida pueda considerarse como Roya dando un diagnóstico errado.

Por otro lado, los modelos basados en el descriptor de Haralick sobre imágenes LBP toman en cuenta tanto textura como color, sin embargo no llegan a alcanzar el porcentaje de acierto de los clasificadores mencionados líneas arriba muy probablemente por la calidad de las imágenes, ya que en muchas de ellas no se distingue la textura de una hoja enferma de otra sana. Aun así, estos modelos también obtuvieron resultados muy precisos, el siguiente modelo con mejores con mejores resultados fue el SVM_HLBP_128x128_HSV_d5 (ACC=89.40%) para muestras con resolución de 128x128. Tal como se explicó en el capítulo 2, el valor de la métrica ACC muestra el porcentaje de instancias correctamente clasificadas.

Dado que las características de textura no se distinguen en imágenes muy pequeñas,se seleccionó el modelo basado en el descriptor de histogramas RGB para muestras de 64x64 y el modelo basado en el descriptor de Haralick para muestras de 128x128.

7 DETERMINACIÓN AUTOMÁTICA DEL GRADO DE SEVERIDAD

El presente capítulo consta de cuatro secciones las cuales se desarrollan de la siguiente manera:

- Pre procesamientos de imágenes (Calibración de contraste, cambio de espacio de colores).
- Segmentación de manchas por el método de Clústering.
- Segmentación de manchas por el método de Thresholding.
- Cálculo de severidad.
- Resultados obtenidos.

7.1 Pre procesamiento de imágenes

Para lograr la independencia del ambiente en el cual se tomaron las fotos, se recurrió a calibrar el contraste; por otro lado, para obtener mejores resultados, se decidió realizar pruebas cambiando el espacio de color de las imágenes.

7.1.1 Calibración de contraste

Para este apartado se desarrolló un script en Python el cual depende de un parámetro al cual denominamos Factor Contraste, éste regula la intensidad del tono de los píxeles de la imagen, al incrementar este parámetro se obtendrá el efecto de oscurecimiento de la imagen, lo que es beneficioso para asimilar el tono de la hoja respecto al fondo (negro). En la Figura 7.1 se muestra la imagen de una hoja a la cual se la ha disminuido el contraste.

Figura 7.1 Imagen con reducción de contraste

7.1.2 Cambio de espacio de colores

Para poder encontrar el espacio de colores más apropiado se escogieron principalmente los espacios de colores Luv y Lab y de ellos los canales ‘u’ y ‘a’ respectivamente, debido a que no son sensibles al brillo, pero mejores resultados entre ellos se obtuvieron para el canal ‘u’ del espacio Luv, el cual se describe a continuación:

- **Espacio de color Luv**

  Este espacio o modelo de color tiene por objetivo ofrecer un espacio perceptualmente uniforme, lo cual quiere decir que la distancia Euclidiana entre dos intensidades de color está fuertemente relacionada con la percepción visual del ser humano (Tkalcic & Tasic, 2003). En la Figura 7.2 se observan las fotografías H033.jpg luego de aplicarle una transformación del espacio RGB al espacio Luv. En la Figura 7.3 se observan los resultados.

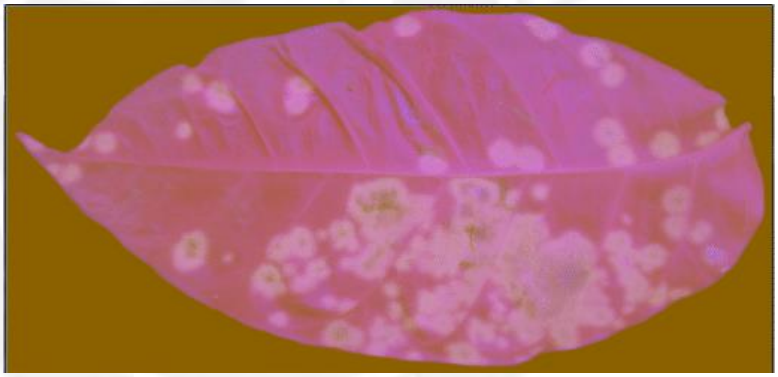

Figura 7.2 Hoja de prueba en espacio de color LUV.

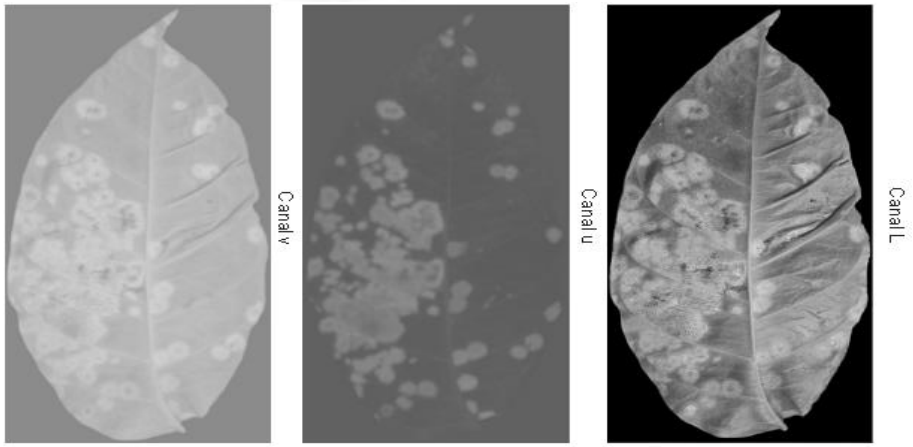

Figura 7.3 Descomposición de Canales del espacio de color LUV.

A partir de los resultados obtenidos se pudo apreciar que en el canal u las manchas de Roya quedaban bastante diferenciadas de lo que es la hoja sana. Este canal sería un excelente candidato para aplicar una técnica de segmentación.

Luego de realizado este análisis se decidió aplicar las técnicas de segmentación elegidas para el espacio de color Luv y canal 'u' respectivo.

Cabe resaltar que aunque se utilizan algunas imágenes como muestra experimental para este documento, la aplicación de los métodos de segmentación y cálculo del índice de severidad se realizan automáticamente para todas las hojas.

7.2 Segmentación de manchas por el método K-means:

Para el desarrollo del programa se utilizó la función k-means (técnica de Clústering que fue explicada en el capítulo 2) provista por la librería de visión computacional OpenCV. La función se define de la siguiente manera (Corporation, 2001):

cv2.kmeans(*samples,nclústers,criteria,attempts,flags*)

- Parámetros de entrada:
  - *samples:* conjunto de características sobre las que se aplicará el método.
  - *nclústers*: números de clústers elegidos a priori.
  - *criteria*: tipo de condición de salida (*type,max_iter,épsilon*)
    - cv2.TERM_CRITERIA_EPS: Termina el algoritmo si se alcanza cierta precisión, *épsilon*.
    - cv2.TERM_CRITERIA_MAX_ITER: Detiene el algoritmo luego de cierto número de iteraciones, *max_iter*.
    - cv2.TERM_CRITERIA_MAX_EPS + cv2.TERM_CRITERIA_MAX_ITER: termina el algoritmo cuando alguna de las dos condiciones se cumple.
  - *attemps*: Número de veces que se ejecutara el algoritmo usando diferentes etiquetados iniciales.
  - *flags*: indican cómo se tomaran los centros iniciales. Hay dos posibles opciones cv2.KMEANS_PP_CENTERS, cv2.KMEANS_RANDOM_CENTERS.

- Parámetros de salida:
    - *compactness*: la suma de la distancia al cuadrado desde cada punto hacia su centro.
    - *labels*: etiquetas con las que cada elemento es marcado “0”,”1”,…
    - *centers*: arreglo que guarda los centros de cada clúster.

En este caso se utilizaron los siguientes valores para los parámetros:

- *samples*: valores de los pixeles en cada canal.
- *nclústers*: 3
- *criteria*:cv2.TERM_CRITERIA_MAX_EPS + cv2.TERM_CRITERIA_MAX_ITER
- *attemps*:10
- *flags*: cv2.KMEANS_RANDOM_CENTERS

El parámetro más importante de todos los elegidos fue el número de *clústeres*. Se eligieron tres puesto que se busca separar en tres regiones la imagen, estas son: el fondo, las regiones sanas de la hoja y las regiones infectadas de la hoja.

En las pruebas iniciales del método se encontró un problema significativo, y es que en cada ejecución la asignación de las etiquetas era aleatoria, es decir que, las manchas de Roya podían ser etiquetadas de color gris en una ejecución y a la siguiente podían ser etiquetadas de color blanco. Esto haría imposible la tarea de realizar un conteo automático del grado de severidad del hongo. Vale aclarar que el término etiqueta hace referencia al color del clúster asignado por el método, puesto que a cada pixel se le asigna el valor del centro del clúster al que pertenece, y la manera de identificar a que clúster pertenece es través de las etiquetas. En la Figura 7.4 se observan los resultados obtenidos en tres ejecuciones distintas.

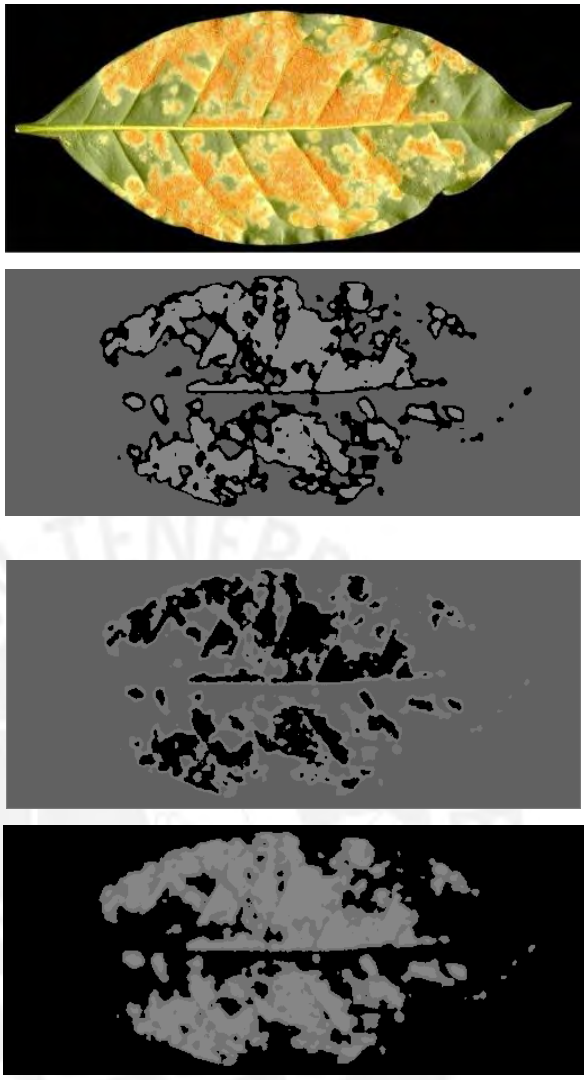

Figura 7.4 Resultados obtenidos por el método Clúster.

Para poder eliminar la aleatoriedad del etiquetado se aprovechó el hecho de que en el canal ‘u’ del espacio de colores Luv, se podía saber a priori la intensidad de los pixeles correspondientes a la Roya Amarilla, a la hoja y al fondo. Para el canal ‘u’, los pixeles asociados a las manchas de Roya tenían la mayor intensidad, le seguían los pixeles asociados al fondo y finalmente se observó que las regiones sanas de la hoja tenían la menor intensidad.

A partir de este análisis a priori, se puede esperar que los centros hallados por el método k-means mantengan relación a la intensidad inicial de los pixeles, puesto que para agruparlos se basa en las distancias euclidianas entre cada uno. De este modo, se desarrolló la siguiente lógica de etiquetado:

- Si el espacio de color es Luv:
    - Si centro[0]>centro[1] y centro[0]>centro[2]
        - centro[0]=255
        - centro[1]=0
        - centro[2]=0
    - Si centro[1]>centro[0] y centro[1]>centro[2]
        - centro[0]=0
        - centro[1]=255
        - centro[2]=0
    - caso contrario
        - centro[0]=0
        - centro[1]=0
        - centro[2]=255

En donde centro[n] es el valor del centro del clúster para n=0,1, 2.

De este modo las manchas de Roya siempre quedan etiquetadas con el valor 255 (blanco puro) y se elimina la aleatoriedad del etiquetado. Sin embargo, esta lógica parte de asumir que la intensidad de las manchas de Roya en cada uno de los canales siempre será la misma, pero esto puede no ser así en todos los casos, ya que hay imágenes con distinta iluminación o manchas de otros colores por lo cual esta es solo una solución aproximada mas no exacta.

Luego de contar con una forma de identificar que clúster correspondía a las manchas de Roya, se procedió a elaborar el script correspondiente. Este programa posee la siguiente lógica:

- Para cada resolución (256, 512,1024):
- Leer imagen
- Para el espacio de color Luv:
    - Aplicar cambio de espacio de color
    - Si es canal Luv:
        - Obtener canal ‘u’ y convertirlo a gris
    - Aplicar método k-means

- Aplicar lógica de etiquetado (clúster Roya = 255)
- Aplicar conteo de pixeles diferentes de cero
- Calcular el porcentaje de severidad

En la figura 7.5 se muestran ejemplos de los resultados obtenidos

Figura 7.5 Resultados obtenidos del clúster binarizados.

7.3 Segmentación por Thresholding

Para el desarrollo de este apartado se empleó el algoritmo de Umbralización o también conocido como "*Thresholding*", se desarrolló en el lenguaje Python y se hizo uso de la librería OpenCV. Con la ayuda de esta técnica se pretende pasar una imagen que se encuentra en escala de grises a otra de solo dos niveles de gris (visualmente reconocidos como blanco y negro).

Por otro lado, para obtener un mejor contraste de las manchas respecto a las hojas se empleó el método de ecualización, con el cual normalizará los niveles de gris de la imagen. Luego, para realizar una correcta umbralización; es decir, el cambio de nivel de grises de 255 a solo dos tipos (negro y blanco) se utilizó el método Otsu, para así encontrar el valor óptimo de umbral con el cual comparar el nivel umbral de cada píxel y cambiarlo según corresponda en comparación con este, y así se obtendrán las imágenes segmentadas en dos niveles de gris los cuales son: blanco (mancha) y negro (fondo).

Para muestra del proceso de Umbralización se tomó una imagen de prueba considerando una buena candidata, debido a que estaba expuesta a brillos, sombras, venas de hojas resaltantes y una cantidad promedio de manchas.

7.3.1 Obstáculos iniciales

Para el cálculo de severidad de la Roya, deben sobrepasarse obstáculos tales como los siguientes:

i. Brillo.
ii. Sombra.
iii. Venas de las hojas.
iv. Aclaración de color debido al tallo de la hoja.

A continuación, se muestran las técnicas empleadas para disminuir el impacto de los obstáculos mencionados.

Como pasos previos se considera que a las imágenes se les ha reducido el contraste; es decir, visualmente se encuentran más oscuras; luego se procederá a cambiar el espacio de color, y como se explicó anteriormente, se cambiará al espacio Luv, y de la descomposición de ellos se empleará el canal **'u'**, ya que ofrece una mejor segmentación

de las manchas respecto a las hojas, con lo cual la imagen (Figura 7.6) queda de la siguiente manera:

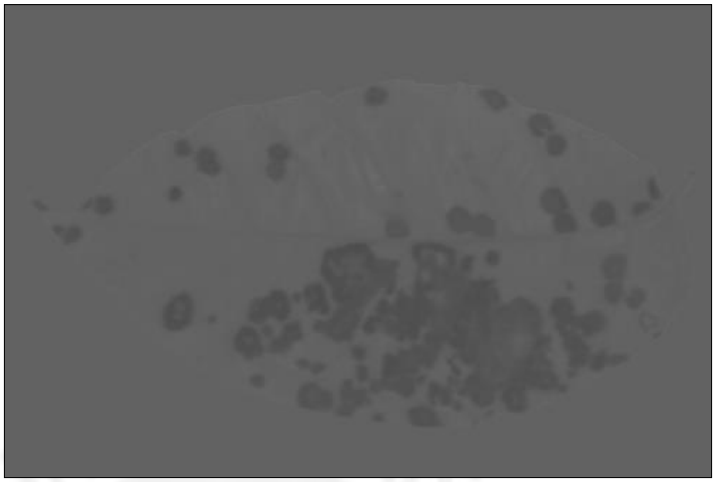

Figura 7.6 Imagen en espacio LUV

Sin embargo, aún para obtener mejores resultados se procedió a ecualizar o también llamado normalizar la imagen con el fin de obtener mejor los balances entre los grises y así resaltar aún más las machas respecto a la hoja. A este proceso último se llama o ecualizar una imagen y consiste en regular los niveles de gris de una imagen; para efectos del proyecto, se busca un mejor contraste entre las manchas respecto a la hoja.

- **Ecualización**

Después que la imagen ha sido tratada previamente y se encuentre en una escala de grises (producto del canal 'u'), se utilizará la técnica de la ecualización. Esta técnica permite que los nivel de gris más obscuros tiendan al negro, y los puntos claros tiendan más al blanco. Para la aplicación de ello, en nuestras imágenes utilizamos una función proporcionada por la librería OpenCV, el cual recibirá como parámetro de entrada la imagen en escala de grises. A continuación se muestra la imagen de prueba (Figura 7.7) sometida a la normalización:

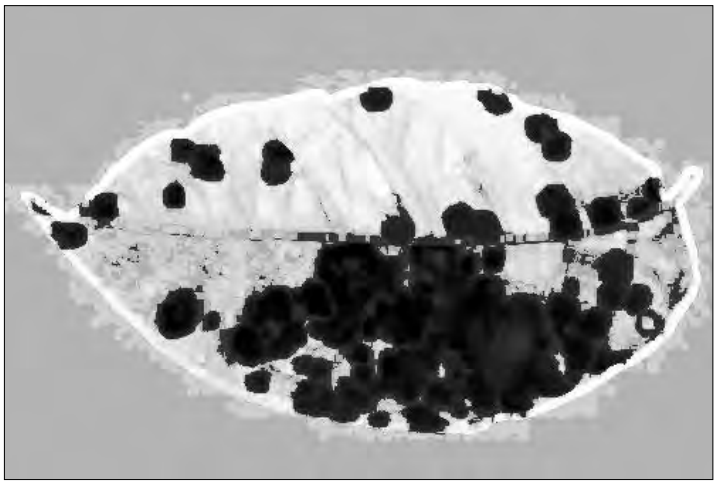

Figura 7.7 Imagen ecualizada.

Como se muestra en la Figura 7.7, hay un mejor contraste entre las manchas y el fondo que representa la hoja misma.

Para notar la diferencia de manera objetiva se muestra el siguiente histograma (Figura 7.8) de la imagen una vez ecualizada:

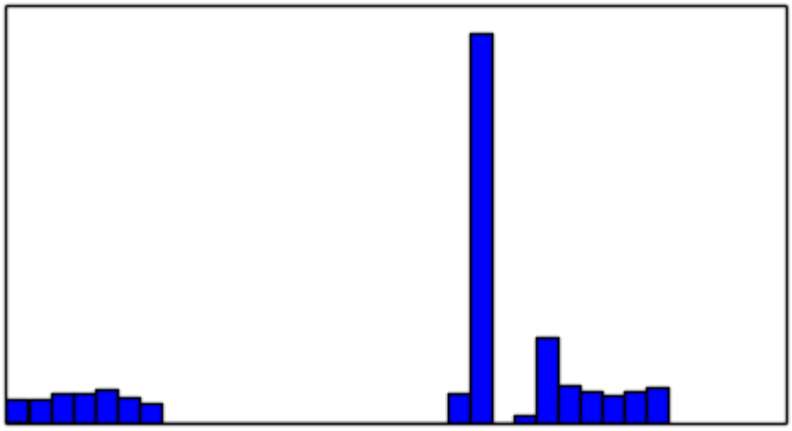

Figura 7.8 Histograma de la imagen ecualizada.

En donde el eje horizontal muestra los niveles de gris, y el eje vertical representa la cantidad de píxeles en aquellos niveles.

7.3.2 Algoritmo desarrollado y uso del método Thresholding

A continuación se muestra el algoritmo para realizar el proceso de umbralización de una imagen:

- Se define los límites del proceso en las variables de recorrido de los ejes vertical y horizontal, estos serán tres diferentes según el tamaño de la resolución que se utilice por muestras, para nuestros casos de análisis, se utilizaron las siguientes escalas horizontales:
    - 256 píxeles
    - 512 píxeles
    - 1024 píxeles

  En cuanto al tamaño para los ejes verticales, estos serán proporcionales al tamaño respectivo de los ejes horizontales para cada muestra respectiva.

- Luego se recorre pixel por pixel la imagen y se va comparando el umbral de cada uno de estos con el umbral óptimo para cada imagen, y se cambia por 0 o 1 según la siguiente regla:

  Si umbralPixel $>$ umbralÓptimo $\rightarrow$ umbralPixel = 0 (negro)

  Si umbralPixel $<$ umbralÓptimo $\rightarrow$ umbralPixel = 1 (blanco)

✓ **Método Otsu**

Para poder hallar el valor umbral óptimo se utilizará este método, ya que es uno de los que mejor resultados provee debido a que trabaja píxel por píxel A continuación se muestra el algoritmo para calcular el valor del umbral óptimo para la binarización de la imagen ("*Thresholding")*se siguió los siguientes pasos:

i. Generar el Vector de Probabilidad acumulada

Para ello se recorrerán todos los píxeles de la imagen en escala de grises y se armará de tal manera que los elementos del vector será el resultado de la cantidad total de píxeles por nivel de gris en el cual se encuentre, estos van desde 0 hasta 255.

ii. Generar el vector de media acumulada

Al igual que el proceso anterior, se formará un vector de medias, donde cuyos elementos contendrán lo mismo que la probabilidad pero con la diferencia que será multiplicado por la escala de bit a la cual pertenezca.

iii. Cálculo de probabilidad acumulada y media acumulada

Los procedimientos anteriores se repetirán tanto el vector de la media como el de la probabilidad tanto para valores de la hoja (fondo) como para el de las manchas (objetos), una vez ello, se halla la varianza y el proceso se repite a lo largo del histograma, quedando de esta manera el valor de umbral óptimo como la varianza máxima en la imagen de grises.

A diferencia del método del K-Means, este método si es automatizado en el cálculo del valor umbral de comparación, lo cual ofrece una mayor ventaja, pues no es un valor estático, sino uno dinámico que se adapta a las condiciones de la imagen. Una vez encontrado el valor umbral para cada imagen se procedió a utilizarlo en el método de Thresholding, ya que cada píxel debe ser comparado con este valor, y actualizado a una nueva escala de grises que comprende 0 ó 1. El resultado para la muestra elegida se ve a continuación (Figura 7.9):

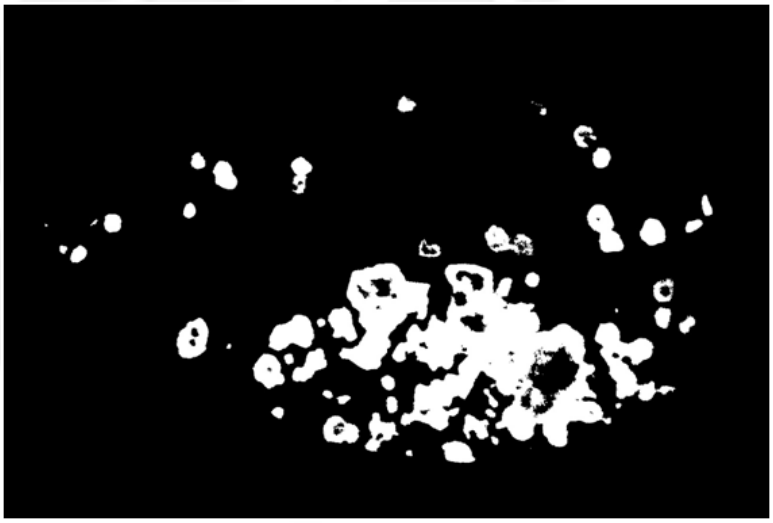

Figura 7.9 Imagen binarizada

De manera objetiva, la distribución de niveles de gris por cantidad de píxeles se puede contemplar en el siguiente histograma (Figura 6.17):

Figura 7.10 Histograma de la imagen binarizada

Donde el eje horizontal representa los niveles de gris que para este caso de la imagen binarizada queda relegada a solo dos niveles 0 (que representa el fondo negro) y 255 (que representa el blanco; es decir, las manchas) y en el eje vertical representa la cantidad de píxeles según nivel.

Finalmente, sobre la imagen binarizada, se aplicó dos operadores morfológicos como Erosión y Dilatación, debido a que los resultados finales se mostraban con información excesiva tales como puntos alrededor de los bordes de la hoja. El objetivo de ello, es adelgazar la imagen lo suficiente como para eliminar los puntos en exceso, y luego restaurar la imagen con el proceso inverso; debido a la dimensión insignificante de los puntos no afecta a la información relevante como las manchas al momento de aplicarlo.

- **Erosión**

  Se le aplica esta operación morfológica con el fin de eliminar ruidos pequeños de la imagen tales como venas de la hoja, puntos en exceso debido a los bordes de la hoja. Ya que lo que realiza esta operación es la eliminación de píxeles del contorno de objetos; es decir, es muy apropiado para el problema que se quiere resolver. El número de iteraciones para las imágenes es 2, puesto que las puntos no tenían una dimensión significativa respecto a las manchas.

- **Dilatación**

  Para la aplicación de esta operación morfológica se selecciona el cada píxel de la imagen, luego se busca el mayor de los píxeles de la vecindad, el cual incluye también el píxel central, y finalmente se reemplaza el valor del píxel seleccionado

por el máximo valor del píxel encontrado de la vecindad. Para efectos de proyecto, resulta muy efectivo, pues una vez que se aplicó la operación de la erosión (iteración igual a 2), se necesita que la imagen recupere su tamaño original, para así no perder información relevante (como las manchas, que son lo que importan), Por lo cual el número de iteraciones para la ejecución de este operador también es 2.

7.4 Cálculo de severidad

Para llevar a cabo este cálculo se desarrolló un script en Python que calcula la cantidad de píxeles con nivel de gris en 255 (blanco) tanto para las manchas, como para su respectiva área total de las hojas; para lograr este último, se realizó aplicó el método del componente Threshold que ofrece la librería OpenCV y a este se utilizó como umbral fijo de comparación un valor bajo igual a quince (15); así como la aplicación de operadores morfológicos (Erosión y Dilatación) para eliminar información excesiva (puntos en los bordes) y así afinar la solución del área de la hoja total. En la figura 7. 11 se aprecia el efecto del área total de la hoja:

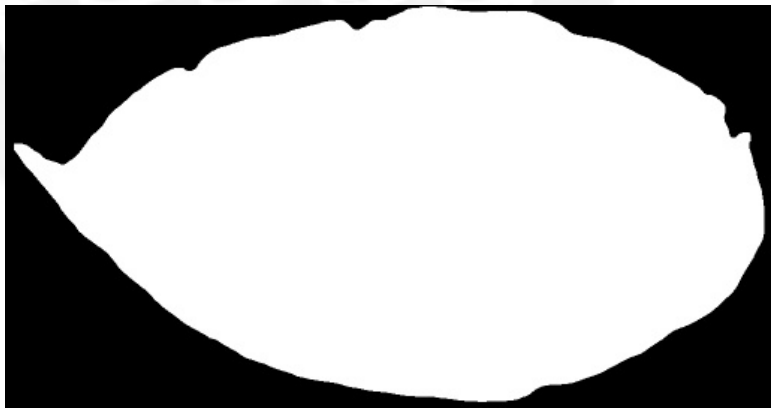

Figura 7.11 Área de la hoja completa.

Finalmente, se calculó el porcentaje de severidad dividiendo la cantidad de píxeles blancos debido a las manchas entre la cantidad total de la hoja.

7.5 Resultados esperados

Para poder comparar de manera equitativa los métodos se ha escogió del método Clústering el que fue producto del cambio de espacio de colores a Luv, ya que según la experimentación mostrada en el apartado 6.2 se mostró un resultado más preciso respecto al método manual.

En la Tabla 7.1 se muestra el resumen de los resultados obtenidos en la comparación de ambos métodos:

Tabla 7.1 Cuadro comparativo de métodos de segmentación

Resolución	Error Promedio Thresholding	Error Promedio K-means	Diferencia Media de Métodos
256x256	8.92%	17.52%	**8.60%**
512x512	5.32%	16.04%	**10.72%**
1024x1024	3.75%	15.90%	**12.16%**

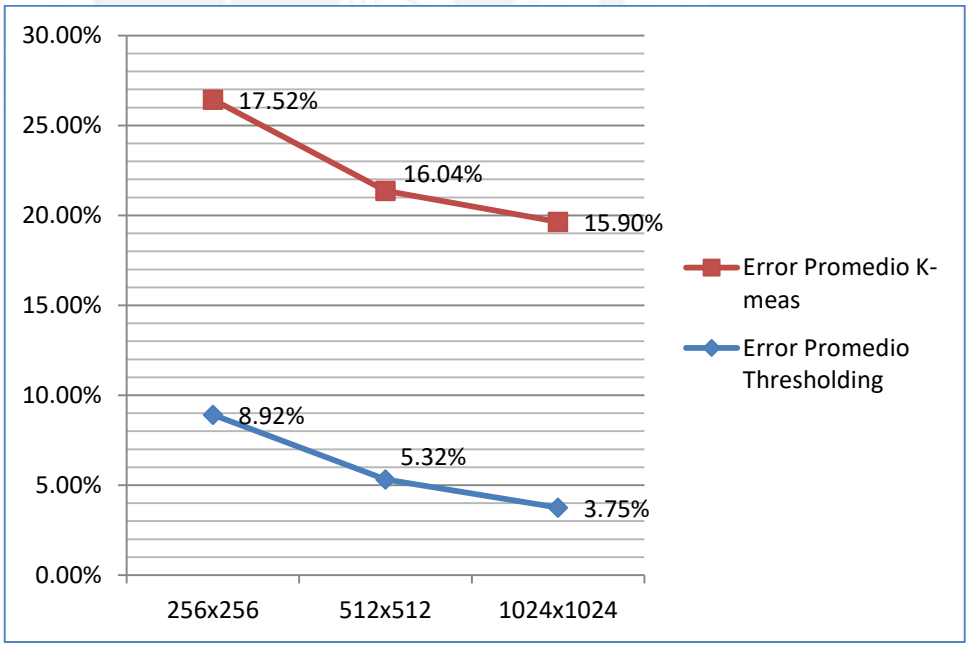

Figura 7.12 Gráfico estadístico de comparación de métodos de segmentación

Como se puede mostrar en la tabla 7.1, los valores promedio de los márgenes de error de los métodos de segmentación respecto al método manual, se puede afirmar que el método por Thresholding es mejor en las tipos de resoluciones, siendo la resolución

óptima la de 1024x1024, pero se optó por la intermedia, esto es la resolución de 512x512, debido a que muestra casi igual valor de diferencia entre el error utilizando el método Thresholding respecto al K-Means (en espacio LUV).

Para verlo de manera más objetiva, se puede considerar la Figura 7.12, en la que se muestra la superioridad del método de Umbralización.

En el Anexo 3, se adjuntan las tablas de comparación de métodos para cada imagen según la resolución de éstas. A continuación se muestra los gráficos estadísticos de comparación por margen de error (Figuras 7.13, 7.14 y 7.15) para cada método de segmentación aplicado a cada tipo de resolución respectivamente:

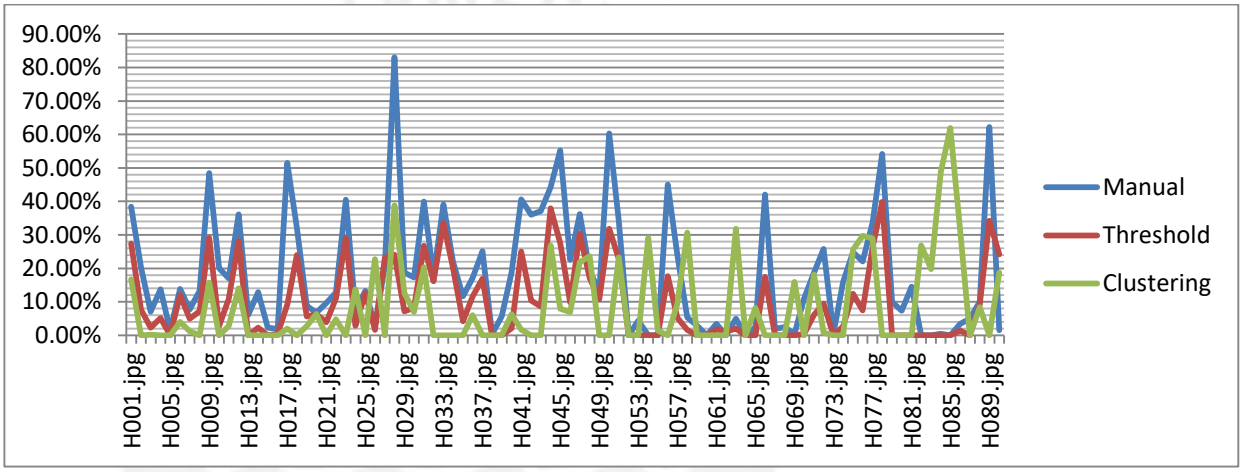

Figura 7.13 Gráfico estadístico de comparación entre los métodos de segmentación para 256x256

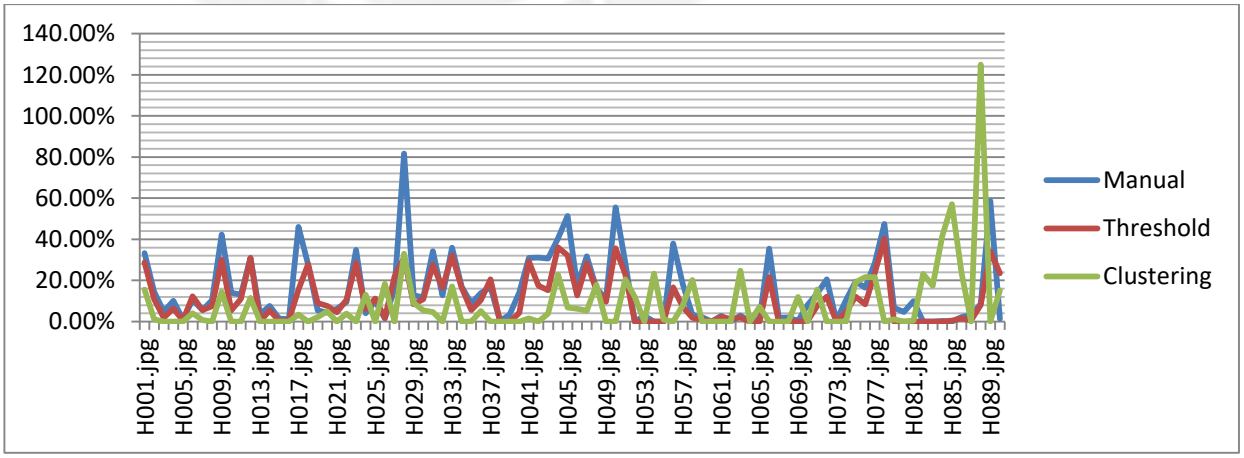

Figura 7.14 Gráfico estadístico de comparación entre los métodos de segmentación para 512x512

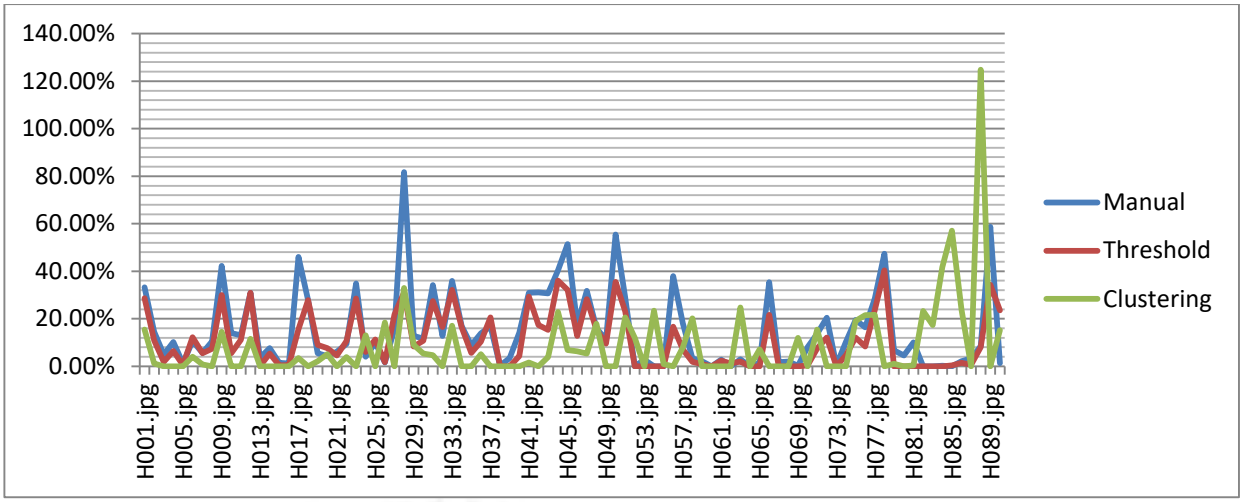

Figura 7.15 Gráfico estadístico de comparación entre los métodos de segmentación para 1024x1024

8 OBSERVACIONES, CONCLUSIONES Y RECOMENDACIONES

8.1 Observaciones

La presente investigación aplicada consideró principalmente conceptos de procesamiento de imágenes y aprendizaje de máquina; además, así como conceptos matemáticos, en especial de algebra lineal puesto que todas las operaciones en imágenes se manejan a través de operaciones en matrices. El pensamiento algorítmico de igual modo fue crucial para el desarrollo de cada etapa. Además, este tipo de proyecto ha permitido tener un mayor acercamiento a lo que es el desarrollo de un proyecto de investigación.

Un hecho importante observado durante el trabajo fue la necesidad de contar con fotografías tomadas en un ambiente debidamente controlado, y evitar posibles mal interpretaciones en los resultados obtenidos por los diferentes métodos probados. Por ejemplo, para el caso de la elección del mejor modelo de clasificación, el que mejor resultados arrojó fue el basado en el descriptor de histogramas RGB pese a ser realmente básico en comparación con el descriptor de Haralick. Hemos observado que esto se debe al hecho de que en todas las muestras no se aprecia bien la textura de manera detallada del hongo (debido a la resolución de la foto), pero sí el color (manchas en contraste con la hoja), quitándole peso al descriptor de Haralick. A pesar de ello, los resultados obtenidos fueron igualmente buenos.

Otro punto que se debe tener en consideración es la cantidad de muestras de las que se dispone. Para la etapa de clasificación la cantidad de datos usados en el entrenamiento del modelo es crucial para obtener buenos resultados. En este proyecto se contó con una cantidad de 90 imágenes; sin embargo, se considera que esta cantidad podría ser ampliada en un futuro para volver a entrenar los modelos y ver si se consiguen mejores resultados.

Se lograron conseguir todos los objetivos planteados, desde la extracción de características de una región sana o infectada hasta la cuantificación del grado de severidad y la implementación de todo el modelo algorítmico en una herramienta grafica funcional.

8.2 Conclusiones

✓ **Relacionadas al objetivo específico 1**

- Se logró reunir un número suficiente de imágenes, para realizar los experimentos y el desarrollo adecuado de la investigación.
- Para el caso de la discriminación automática de una hoja infectada de una sana, no se contaba con suficiente cantidad de fotografías de hojas sanas (menor a 30 imágenes), por lo cual se optó por segmentar la imagen en partes más pequeñas y realizar la evaluación por regiones, aumentando de esta forma el número de muestras disponibles. Esta idea permitió continuar con la investigación y además trajo buenos resultados.

✓ **Relacionadas al objetivo específico 2**

- Se logró implementar en Python el operador de Patrones Binarios Locales LBP (del inglés *Local Binary Patterns*) modificado para tomar en consideración variaciones de color.
- Se logró implementar en Python el descriptor de textura de Haralick, lo cual permitió procesar las imágenes y extraer de ellas las características necesarias para el etapa de clasificación
- Se logró implementar un descriptor basado únicamente en el histograma RGB de la imagen, es decir en la intensidad de los pixeles en cada canal de color.
- Al texturizar las imágenes con el operador LBP las manchas de Roya quedaron mejor resaltadas en la mayoría de los casos.

- El aplicar primero el operador LBP y luego usar el descriptor de Haralick sobre la imagen texturizada incremento el tiempo de procesamiento.
- El descriptor basado en histogramas RGB proceso las imágenes y extrajo las características en un tiempo mucho más corto al que le tomo al descriptor de Haralick.
- Se generaron diez conjuntos de datos, variando la configuración del descriptor de Haralick, el espacio de color sobre el cual se aplicaban los métodos y la resolución de las muestras.

✓ **Relacionadas al objetivo específico 3**

Se entrenaron diez modelos con los diez conjuntos de datos diferentes, todos basados en el algoritmo de aprendizaje de máquina conocido como Máquinas de Vectores de Soporte SVM (del inglés *Support Vector Machine*)

- Cuatro de los diez modelos obtuvieron una exactitud en la clasificación superior al 80%, superando el objetivo propuesto.
- La calibración previa de los parámetros del algoritmo SVM fue crucial para la obtención de tan buenos resultados.

✓ **Relacionadas al objetivo específico 4**

- Los dos mejores modelos (87.50% y 95.53%) fueron los entrenados a partir de las características extraídas con el descriptor RGB; sin embargo, este puede ser un resultado engañoso pues solo toma en consideración la variación en la intensidad de los pixeles en cada canal de color.
- El tercer mejor modelo obtuvo una exactitud del 89.40% en muestras de 128x128 de resolución a partir del descriptor de Haralick sobre el espacio de color HSV, con una separación *d=5* entre pixeles de referencia y pixeles vecinos.
- Para los modelos basados en el descriptor de Haralick, en general se obtuvieron mejores resultados de clasificación en las muestras de 128x128 de resolución, sobre el espacio HSV y a una distancia de separación de cinco entre pixel vecino y de referencia.
- Por lo discutido líneas arriba, se podría considerar como mejor descriptor al de Haralick sobre el espacio de color HSV con una separación entre pixeles *d=5*.

- ✓ **Relacionadas al objetivo específico 5,6 y 7**
  - Se utilizó el método de segmentación por Clustering conocido como k-means, pero además se agregó una lógica de etiquetado que permite identificar cual fue el clúster asignado a las manchas de Roya; aún con todo no tuvo buenos resultados debido a que le afectó los cambios de tono de la hoja debido a los diferentes ambientes en los cuales fueron fotografiados.
  - El mejor espacio de color para trabajar la segmentación por k-means fue el espacio Luv por sobre el espacio de colores Lab, y de éstos el mejor resultado canal a utilizarse fue el 'u' respecto al 'a', debido a que el primero filtraba mejor los brillos de las imágenes.
  - En hojas completamente sanas, o con severidad menor al 10%, el método es totalmente impreciso, con lo cual se hace necesaria la participación de una clasificación previa de las imágenes en hojas sanas e infectadas.
  - Se obtuvieron resultados muy similares en todas las resoluciones probadas (256, 512 y 1024), por lo que se concluye que el método funcionaria bien sin importar la resolución de la cámara con la cual se tome la fotografía. Sin embargo, se debe tener en consideración que a mayor resolución, mayor es el tiempo de procesamiento pero esto no garantiza mejores resultados. Aún con todo, los resultados que arroja el K-means no son del todo precisos, ya que habían crasos errores en la segmentación de manchas, pues tomaba a toda la hoja como si fuera una gran mancha blanca, en cambio el del Thresholding, sí tiene un buen resultado medianamente diferenciado entre resoluciones, del cual se concluye que a mayor resolución mayor precisión; pero por consto-oportunidad, nos quedamos con la resolución intermedia de 512x512 píxelesxpíxeles.
  - El método Thresholding ofrece una mejor precisión en los resultados debido a que no le afecta el brillo, debido al espacio de colores en el que se realizó el experimento (Luv) y de ellos el canal 'u'.
  - La calibración de la ecualización fue el factor determinante para el éxito del método Thresholding respecto a K-means, pues se llegó a un equilibrio de los resultados en la segmentación de las manchas, sin que le afecte en gran medida la luz sobre la imagen (brillo), ni tampoco las sombras; así también, la baja o alta presencia de Roya no afectó significativamente a la segmentación por medio del método Thresholding, por lo cual fue contundente su superioridad en precisión respecto al K-means.

- La reducción del contraste, fue un factor crítico para la mejores de los resultados para ambos métodos de segmentación, ya que casi alineaba el tono de la hoja respecto al fondo negro de la imagen.
- La diferencia de los resultados obtenidos de cálculo de severidad del mejor método (Threshold) es relativamente pequeño con respecto a la segmentación de las manchas realizada de manera manual, esto es consistente debido a que hay un factor subjetivo por parte de la persona que segmenta las manchas manualmente, por la falta de precisión del usuario en este proceso engorroso. Si se observan los resultados obtenidos, el error es por defecto y no por exceso respecto a la segmentación manual.

✓ **Relacionadas al objetivo específico 8**

- La pequeña aplicación permitirá que el usuario tenga un reporte con la información de la severidad por fotografía de las hojas y la respectiva clase de infección a la cual pertenece.
- La aplicación ofrecerá al usuario una clasificación de las imágenes categorizándolas en diferentes carpetas una vez ejecutada la segmentación automatizada por medio del uso de la herramienta.

8.3 Recomendaciones

- El proyecto puede escribirse también en lenguaje de programación C y C++, debido a que la librería del OpenCV también es compatible con el ese lenguaje, y ofrece una gran cantidad de algoritmos de Visión computacional y documentación en C y C++ (Coporation Intel, 2001).
- Para investigaciones y proyectos similares se debe contar de ante mano con una base datos de imágenes suficientemente amplia para realizar todos los experimentos planeados dentro del cronograma.
- Se debe tener en consideración el tiempo que toma procesar las imágenes y extraer las características para no demorar el cronograma de trabajo.
- En futuros trabajos se puede probar combinando las características extraídas por el descriptor de Haralick y por el descriptor de histogramas RGB.
- En futuros trabajos se puede contrastar los resultados obtenidos por otros algoritmos de aprendizaje de máquina como Naive Bayes, K-vecinos más cercanos o una red neuronal.

- Al contar con un mayor número de muestras que las utilizadas en este proyecto, resulta más provechoso probar la construcción variando los tamaños de los conjuntos de entrenamiento.
- Se podrían construir y probar más modelos variando los espacios de color y aumentando el número de distancias *d*.

REFERENCIAS BIBLIOGRÁFICAS

Aguilar Carrera, G. G. (1995). Procesamiento digital de imágenes utilizando filtros morfológicos (Doctoral dissertation, QUITO/EPN/1995).

Al Bashish, D., Braik, M., & Bani-Ahmad, S. (2010, 15-17 Dec. 2010). A framework for detection and classification of plant leaf and stem diseases. Paper presented at the Signal and Image Processing (ICSIP), 2010 International Conference on.

Anthonys, G., & Wickramarachchi, N. (2009, 28-31 Dec. 2009). An image recognition system for crop disease identification of paddy fields in Sri Lanka. Paper presented at the Industrial and Information Systems (ICIIS), 2009 International Conference on.

Arivazhagan, S., Shebiah, R. Newlin, Ananthi, S., & Varthini, S. Vishnu. (2013). Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features. Agricultural Engineering International: CIGR Journal, Vol. 15(1), 211.

Avelino, Jacques, Muller, Raoul, Eskes, Albertus, Santacreo, Rodney, & Holguín, Francisco. (1999). La Roya anaranjada del cafeto: mito y realidad. Desafios de la caficultura en Centroamérica. San José, CR: IICA/PROMECAFE/CIRAD/IRD, 227-233.

Belkacem-Boussaid, K., Sertel, O., Lozanski, G., Shana'aah, A., & Gurcan, M. (2009, 3-6 Sept. 2009). Extraction of color features in the spectral domain to recognize centroblasts in histopathology. Paper presented at the Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE.

Bishop, C.M. (2006). Pattern recognition and machine learning: Springer.

Cambronero, C. G., & Moreno, I. G. (2006). Algoritmos de Aprendizaje: knn & kmeans. Intelgencia en Redes de Comunicación, Universidad Carlos III de Madrid.]

Freund, Yoav, & Schapire, Robert E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1), 119-139.

Golzarian, M. (2011, 26-28 July 2011). Adaptive segmentation of plant images, an integration of color space features and self-organizing maps. Paper presented at the Multimedia Technology (ICMT), 2011 International Conference on.

Gonzales, Rafael C., & Woods, Richard E. (2008). Digital Image Processing (Third Edition ed.). United States of America: Pearson Education, Inc.

Haiguang, Wang, Guanlin, Li, Zhanhong, Ma, & Xiaolong, Li. (2012, 29-31 May 2012). Image recognition of plant diseases based on principal component analysis and neural networks. Paper presented at the Natural Computation (ICNC), 2012 Eighth International Conference on.

Hitimana, Eric, & Gwun, Oubong. (2014). Automatic Estimation of Live Coffee Leaf Infection based on Image Processing Techniques. arXiv preprint arXiv:1402.5805.

Kobayashi, T., & Otsu, N. (2009, 19-24 April 2009). Color image feature extraction using color index local auto-correlations. Paper presented at the Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on.

López, D. (2010). Efecto de la carga fructífera sobre la Roya (Hemileia vastatrix) del café, bajo condiciones microclimáticas del sol y sombra en Turrialba, Costa Rica. (Magister Scientiae Master), Escuela de Postgrado del CATIE.

Marsland, Stephen. (2009). Machine Learning: An Algoritmic Perspective. United States of America: CRC Press LLC.

Rayner, RW. (1972). Micología, historia y biología de la Roya del cafeto.

Rong, Zhou, Kaneko, S., Tanaka, F., Kayamori, M., & Shimizu, M. (2013, 4-6 Dec. 2013). Early Detection and Continuous Quantization of Plant Disease Using Template Matching and Support Vector Machine Algorithms. Paper presented at the Computing and Networking (CANDAR), 2013 First International Symposium on.

Russ, Jhon C. (1999). The Image Processing Handbook (Third Edition ed.). United States of America: CRC Press LLC.

Shih, P., & Chengjun, Liu. (2005, 16-18 Aug. 2005). Extracting efficient color features for face recognition using evolutionary computation. Paper presented at the Computational Intelligence and Multimedia Applications, 2005. Sixth International Conference on.

Sonka, Milan, Hlavac, Vaclav, & Boyle, Roger. (2008). Image Processing, Analisys, and Machine Vision (Third Edition ed.): Thomson.

Soo-Chang, Pei, & Ching-Min, Cheng. (1999). Color image processing by using binary quaternion-moment-preserving thresholding technique. Image Processing, IEEE Transactions on, 8(5), 614-628. doi: 10.1109/83.760310

Wei-Ta, Chen, Wei-Chuan, Liu, & Ming-Syan, Chen. (2010). Adaptive Color Feature Extraction Based on Image Color Distributions. Image Processing, IEEE Transactions on, 19(8), 2005-2016. doi: 10.1109/TIP.2010.2051753

Zhao, Zhang, Mingbo, Zhao, Bing, Li, & Peng, Tang. (2013, 4-9 Aug. 2013). ColorPCA: Color principal feature extraction technique for color image reconstruction and recognition. Paper presented at the Neural Networks (IJCNN), The 2013 International Joint Conference on.

Coporation, Intel. (2001). OpenCV reference manual.

Finlayson, Graham D, Schiele, Bernt, & Crowley, James L. (1998). Comprehensive colour image normalization Computer Vision—ECCV'98 (pp. 475-490): Springer.

Haralick, Robert M, Shanmugam, Karthikeyan, & Dinstein, Its' Hak. (1973). Textural features for image classification. Systems, Man and Cybernetics, IEEE Transactions on(6), 610-621.

Hsu, Chih-Wei, Chang, Chih-Chung, & Lin, Chih-Jen. (2003). A practical guide to support vector classification.

Kohavi, Ron. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Paper presented at the IJCAI.

Kohl, Matthias. (2012). Performance measures in binary classification. International Journal of Statistics in Medical Research, 1(1), 79-81.

MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

Manjunath, Bangalore S, Ohm, J-R, Vasudevan, Vinod V, & Yamada, Akio. (2001). Color and texture descriptors. Circuits and Systems for Video Technology, IEEE Transactions on, 11(6), 703-715.

Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1), 62-66.

Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, . . . Dubourg, Vincent. (2011). Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825-2830.

Porebski, Alice, Vandenbroucke, Nicolas, & Macaire, Ludovic. (2008). Haralick feature extraction from LBP images for color texture classification. Paper presented at the Image Processing Theory, Tools and Applications, 2008. IPTA 2008. First Workshops on.

Powers, David Martin. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.

Sokolova, Marina, & Lapalme, Guy. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427-437.

Tkalcic, Marko, & Tasic, Jurij F. (2003). Colour spaces: perceptual, historical and applicational background. Paper presented at the Eurocon.