DISEÑO DE UN SISTEMA DE VISIÓN ARTIFICIAL DE BAJO COSTE PARA EL CONTROL FITOSANITARIO DE CULTIVOS DE LULO

ING. JOHN JAIRO CASTRO MALDONADO

MAESTRÍA EN CONTROLES INDUSTRIALES
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
VICERRECTORIA DE INVESTIGACIONES
UNIVERSIDAD DE PAMPLONA
2018

DISEÑO DE UN SISTEMA DE VISIÓN ARTIFICIAL DE BAJO COSTE PARA EL CONTROL FITOSANITARIO DE CULTIVOS DE LULO

Memoria Del Trabajo De Investigación Para La Obtención Del Título Magíster En Controles Industriales

AUTOR
ING. JOHN JAIRO CASTRO MALDONADO

DIRECTOR:
OSCAR EDUARDO GUALDRON GUERRERO, Ph.D

CODIRECTOR
JAIME ALBERTO ECHEVERRI ARIAS, Ph.D

MAESTRÍA EN CONTROLES INDUSTRIALES
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
VICERRECTORIA DE INVESTIGACIONES
UNIVERSIDAD DE PAMPLONA
2018

TABLA DE CONTENIDO

Pág.

1. INTRODUCCIÓN 4

1.1 Motivación 4

1.2 Hipótesis. 7

1.3 Objetivos. 7

1.3.1 Objetivo General. 7

1.3.2. Objetivos Específicos. 7

1.4. Metodología (Diseño Del Experimento) 7

1.5. Organización de la memoria 9

2. MARCO TEÓRICO Y ANTECEDENTES 11

2.1. Introducción del Capítulo 11

2.2. Cultivo del lulo 12

2.3 Factores fitopatógenos del lulo 15

2.3.1.1. Tizón del lulo o gota (Phytophthora infestans) 16

2.3.1.2 Moho blanco, lama blanca, pudrición algodonosa 16

2.3.1.3. Antracnosis del fruto 17

2.3.1.4. Pudrición del tallo por Esclerotium 18

2.3.1.5 Marchitez vascular (Fusarium oxysporum) 19

2.3.1.6. Agalla radical (Meloidogyne spp) 20

2.3.1.7. Alternaria 21

2.4. Agricultura de Precisión (A.P) 23

2.5. Visión Artificial 38

2.5.1. Introducción 38

2.5.2. Definición 39

2.5.3. Elementos de los sistemas de visión artificial 40

2.5.3.1. Sistema de iluminación 40

2.5.3.2. Cámaras y Tarjeta de Captura 42

2.5.3.3. Hardware y software 42

2.5.3.3.1 Hardware 42

2.5.3.3.2 Software 42

2.5.4. Etapas 45

2.5.4.1. Adquisición 45
2.5.4.2. Procesamiento 51
2.5.4.3. Segmentación 60
2.5.4.4. Extracción de características 67
2.5.4.5. Reconocimiento e interpretación 68
2.5.5.1. Python 70
2.5.5.2. Open CV 72
3. DESARROLLO DEL SISTEMA DE DETECCIÓN 74
3.1 HARDWARE Y PERIFÉRICOS Y/O ACCESORIOS A UTILIZAR 74
3.2 Instalación del sistema operativo y puesta en marcha de la Raspberry Pi 2 84
3.3 Implementación, puesta en funcionamiento y verificación de aplicaciones o programas similares encontrados en la literatura y la web 85
3.3.1 Detección Facial 86
3.3.2 Detección de mano abierta 90
3.3.3 Detección de Colores 96
3.3.4 Detección de Círculos 102
3.3.5 Comparación de los algoritmos analizados encontrados en la literatura 106
3.3.6 Desarrollo del algoritmo de detección de Manchas de “Alternaria” en tiempo real 107
3.3.6.1 Procedimiento a partir de la segmentación e identificación de figura geométricas 107
3.3.7 Procedimiento con clasificadores en Cascada basados en características Haar y HOG 110
4. PRUEBAS Y VERIFICACIONES 126
5. CONCLUSIONES 134
REFERENCIAS BIBLIOGRAFICAS 136

LISTA DE FIGURAS

Pág.

Figura 1. Metodología del trabajo de investigación....8
Figura 2. Proceso de verificación del sistema....9
Figura 3. Estructura de la memoria....10
Figura 4. Fruto del lulo....12
Figura 5. Cultivo de lulo en la región de Antioquia....13
Figura 6. Producción de lulo en Colombia....13
Figura 7. Phytophthora en lulo....16
Figura 8. Presencia en el tallo de moho blanco....17
Figura 9. Antracnosis en tallo – Peñon (Cund)....18
Figura 10. Pudrición del tallo por Esclerotium....19
Figura 11.Corte transversal de tallo, mancha café evidencia de fusarium....20
Figura 12. Amarillamiento, síntoma de presencia nemátodos....21
Figura 13. Patología de Mancha de Alternaria....22
Figura 14. Mancha de Alternaria en la Finca “La Corraleja”....23
Figura 15. Técnicas de Agricultura de Precisión (A.P)....33
Figura 16. Implementación de Tecnologías en Industrias de España....35
Figura 17. Anchura de banda requerida Vrs Capacidad de rango de distancia corta, celular y LPWA....38
Figura 18. Niveles de Procesamiento de imágenes....39
Figura 19. a) Iluminación posterior, b) Frontal, c) Direccional y d) Estructurada....42
Figura 20. Software y sistemas operativos usados en las RASPBERRY Pi....43
Figura 21. Distribución de librerías dentro del paquete de Open CV....45
Figura 22. Proceso de Adquisición de imagen digital....46
Figura 23. Espectro electromagnético....47
Figura 24. Generación de una imagen digital.....48
Figura 25. (a) Imagen original (b) Imagen continúa proyectada sobre un arreglo de sensores. (c) El resultado del muestreo y cuantificación de imagen....49
Figura 26. Representación de los pixeles 8 - vecinos....51
Figura 27. Operaciones aritmético – lógicas....52
Figura 28. Operaciones Geométricas de una imagen....52
Figura 29. Imagen original (a) frente a imágenes con ajuste de contraste (b) y (c)....53
Figura 30. Imagen original (a) frente a imagen con el histograma ecualizado (b)....54
Figura 31. Imagen en escalas de grises Vrs imagen ecualizada....54
Figura 32. Etapas del proceso de convolución para el filtrado en el dominio de la frecuencia....55
Figura 33. Tipos de filtros en el dominio de la frecuencia.....56
Figura 34. Imagen en escala de grises Vrs imagen suavizada....57
Figura 35. Imagen original Vrs imagen con filtro de suavizado....57
Figura 36. Imagen original (Izquierda) frente a imagen tratada con el filtro de Canny (Centro) y tratada con el filtro de Sobel (derecha)....58
Figura 37. Operaciones Morfológicas que se pueden realizar a una imagen....60
Figura 38. Etiquetado de los objetos de una imagen....61
Figura 39. Imagen en escala de gris Vrs imagen binarizada....62
Figura 40. Umbralización de Otsu....63
Figura 41. Raspberry Pi 2 Modelo B....75

Figura 42. Comparación de Raspberry Pi con la competencia 75
Figura 43. Tarjeta Micro SD usada en el desarrollo 77
Figura 44. Cargador Portátil de dispositivo móvil 77
Figura 45. Dongle Wifi Raspberry Pi 78
Figura 46. Webcam comercial 79
Figura 47. Raspicam 79
Figura 48. Luxómetro HS1010A Usado para la validación del algoritmo de visión artificial 81
Figura 49. Cámara fotográfica profesional usada en el proyecto 82
Figura 50. Carga de la imagen Raspbian a la SD con el software Win 32 84
Figura 51. Etapas de configuración del S.O Raspbian en la Raspberry Pi 85
Figura 52. Código en Python para el reconocimiento facial 87
Figura 53. Código de captura del programa de reconocimiento facial 88
Figura 54. Trabajo experimental realizado al algoritmo de reconocimiento facial aplicando clasificadores en cascada tipo Haar 89
Figura 55. Resultados del trabajo experimental del algoritmo de reconocimiento facial con clasificadores cascada tipo Haar 90
Figura 56. Defectos de convexidad del contorno de la mano 91
Figura 57. Tipos de Umbralización aportados por OpenCV 92
Figura 58. Código para identificar apertura de la mano usando reconocimiento de patrones a través de rutinas orientadas al cálculo geométrico 94
Figura 59. Validación del algoritmo de rutinas orientadas al cálculo geométrico 95
Figura 60. Resultados del comportamiento de detección del algoritmo orientado a rutinas de cálculos geométricos 95
Figura 61. Código clasificador de Colores 98
Figura 62. Ensayo de detección de colores con una tolerancia de +- 20 99
Figura 63. Comportamiento de detección del algoritmo de detección de colores con tolerancia de +- 20 .99
Figura 64. Figura 64. Ensayo de detección de colores con una tolerancia de +- 30 100
Figura 65. Comportamiento de detección del algoritmo de detección de colores con tolerancia de +- 30 100
Figura 66. Ensayo de detección de colores con una tolerancia de +- 40 niveles de RGB 101
Figura 67. Comportamiento de detección del algoritmo de detección de colores con tolerancia de +- 40 101
Figura 68. Consolidación de los resultados de la detección de colores con las tres tolerancias estudiadas. 102
Figura 69. Código de detección de figuras circulares en la imagen 103
Figura 70. Validación de algoritmo de detección de figuras geométricas circulares 105
Figura 71. Figura 71. Resultados de detección del código de detección de figuras circulares 105
Figura 72. Consolidad de resultados de detección de los diferentes códigos estudiados 106
Figura 73. Código de combinación de detección de figuras circulares y colores específicos 109
Figura 74. Kernel Haar para extracción de características de las imágenes 110
Figura 75. Descriptores HOG sobre una imagen. 111
Figura 76. Esquema de un clasificador en cascada 112
Figura 77. Muestras de imágenes positivas para el entrenamiento 114
Figura 78. Muestras de imágenes negativas para el entrenamiento 115
Figura 79. Archivo de Texto con características de las imágenes negativas 116
Figura 80. Archivo de texto con el nombre de las imágenes positivas y ubicación del objeto 116
Figura 81. Archivo xml del detector encontrado a través de la segmentación de las imágenes positivas y negativas disponibles 117
Figura 82. Implementación del archivo xml en el programa desarrollado en Python 118
Figura 83. Etiquetado de imágenes usando la APP training image labeler de Matlab (Fuente: Propia)...118
Figura 84. Script de entrenamiento en Matlab usando el algoritmo de Adaboost 119

Figura 85. Figura 85. Script de entrenamiento en Matlab usando el algoritmo de Adaboost.....119
Figura 86. Código de prueba del algoritmo Adabost desarrollado en Matlab.....120
Figura 87. Imagen con etiqueta de detección generada por el código Adabost en Matlab.....120
Figura 88. Código de validación desarrollado en Matlab del algoritmo de entrenamiento Adaboost.....121
Figura 89. Ejemplo de detección de una señal de tránsito y su determinación del caso de efectividad del algoritmo.....122
Figura 90. Imagen con el recuadro de entrenamiento y el recuadro de detección.....123
Figura 91. Intervalos de medición para la evaluación de los detectores 124
Figura 92. Casos extremos de detección.....124
Figura 93. Proceso de validación y pruebas de detección del algoritmo de segmentación 127
Figura 94. Datos de detección del algoritmo de segmentación 127
Figura 95. Proceso de validación y pruebas del algoritmo de detección Adaboost con descriptores HOG 128
Figura 96. Datos de detección del algoritmo Adaboost usando descriptores HOG 129
Figura 97. Proceso de validación y pruebas del algoritmo Adaboost usando filtros tipo Haar 129
Figura 98. Figura 98. Gráfica del porcentaje de detección del algoritmo Adaboost usando filtros tipo Haar 130
Figura 99. Comparación del porcentaje de detección de los tres algoritmos estudiados.....130
Figura 100. Imagen de las detecciones realizadas por el código HOG en Matlab.....131
Figura 101. Imágenes de la comparación de la detección y el entrenamiento del algoritmo en Matlab ...132
Figura 102. Resultados cuantitativos de las pruebas y validación del algoritmo en Matlab.....133

LISTA DE TABLAS

Pág.

Tabla 1. Cuadro comparativo de trabajos relevantes al área de estudio 31
Tabla 2. Tabla 2. Especificaciones de la cámara fotográfica profesional EOS Rebel T3.....82

RESUMEN

El presente trabajo de investigación se enmarca en el área de instrumentación y sensórica específicamente en la visión artificial o por computador, enfocando su aplicación hacia el reconocimiento de patrones para identificar señales visuales que generan algunos factores fitopatógenos de los cultivos de lulo.

Los sistemas tradicionales de producción tratan las propiedades agrícolas y al cultivo en general, de forma homogénea, tomando como base las condiciones promedio de las extensas áreas de producción, para implementar las acciones correctivas o preventivas de los factores identificados.

Con el fin, de obtener unos sistemas de producción más competitivos y aumentar la eficiencia agronómica del sector productivo, se incorporaron nuevas técnicas para incrementar y/o mantener la productividad de los cultivos, buscando, al mismo tiempo, reducir los costos de producción. En ese contexto, la optimización en la detección y ubicación temprana de los factores fitopatógenos que se puedan presentar, es una alternativa importante ya que establece una manera diferenciadora en el manejo del sistema de producción, buscando promover la estabilidad a través de la maximización del retorno económico y preservando el medio ambiente aplicando tecnologías de punta que reduzcan y optimicen los tiempos de inspección [1].

La investigación, se basa en las áreas de visión artificial y reconocimiento de patrones implementado en procesos de agricultura. Con el fin, de desarrollar metodologías, sistemas y/o prototipos que contribuyan a optimizar recursos (humanos, económicos y tecnológicos) en el sector agro y específicamente en las agroindustrias cultivadoras de lulo en Antioquia.

El lulo, *Solanum quitoense Lam*, es el tercer cultivo de mayor relevancia en la población campesina antioqueña, debido a que representa el 1,7% de la producción anual de frutas en Colombia, en este sentido se hace necesario, implementar estrategias novedosas y eficaces que permitan la identificación de algunos fitopatógenos que puedan afectar la producción del cultivo, uno de los más relevantes es la *alternaría Nees* y se evidencia a través de la mancha de la alternaría, esta enfermedad evidencia síntomas como: lesiones redondas de bordes irregulares de color café oscuro o castaño rodeadas de un halo colorítico [2][3][4].

Estas características particulares de algunos fitopatógenos, se pueden aprovechar para incursionar en sistemas de visión artificial que se puedan aplicar a equipos aéreos no tripulados a los que a partir de ahora llamaremos UAS (Unmanned Aerial Systems) y que también se

encuentran en la literatura como UAV *(Unmanned Aerial Vehicle)* y normalmente llamados DRONE [5][6].

La visión por computador o artificial es uno de los campos de la inteligencia artificial, en el que se busca por medio de lenguajes de programación, que un procesador sea capaz de identificar y realizar ciertas ordenes desde el reconocimiento de ciertas características de imágenes tomadas vías fotografía o a través de video [5].

Los sistemas de visión artificial se componen de una fuente de luz que puede ser artificial o natural dependiendo de la aplicación, un sensor de imagen como puede ser una cámara fotográfica o de video, que captura las imágenes que luego se van a procesar, una tarjeta de adquisición de imágenes, como interfaz entre el sensor y el computador o procesador, unos algoritmos de análisis, en los que se procesan, segmentan y se aplican las transformaciones necesarias para obtener los resultados deseados que pueden ser trabajados en múltiples software de procesamiento de imágenes dentro de lo que tenemos las librerías de OpenCV que se trabajan en los software C, C++ y Python, un procesador que analice las imágenes recibidas ejecutando los algoritmos diseñados y un sistema de respuesta en tiempo real que ejecute los comandos [5][6][7][8][9].

Los sistemas de visión artificial de bajo costo desarrollados con sistemas embebidos junto con lo UAS, se consolidan como una de las alternativas más eficaces a la hora de implementar tecnologías de agricultura de precisión, la agricultura de precisión, se puede entender como la implementación de tecnologías o técnicas que optimizan los recursos (humanos, económicos y tecnológicos) en la producción agrícola, con base a ciertos factores o características que pueden ser: variabilidad de características físico-químicas del terreno, condiciones ambientales o características del cultivo en sí [10][11][12].

A través del desarrollo de este trabajo se plantea un sistema de visión artificial de bajo costo para el control fitosanitario de cultivos de lulo usando como base la visión por computador y la inteligencia artificial (IA).

Dentro de los principales resultados están, que efectivamente con un hardware y software libre de bajo costo es viable la elaboración de este tipo de sistemas, ya que en este estudio se puede evidenciar que con aplicaciones como Python y las diversas técnicas que en la actualidad se disponen para realizar estrategias de inteligencia artificial es posible identificar señales de presencia de factores fitopatógenos en el cultivo del lulo, ya que este estrategia permite identificar

o reconocer patrones a 40 o 60 cm de distancia con una base de entrenamiento entre 200 y 400 imágenes, que es factible para su aplicación en la agricultura y en vehículos aéreos no tripulados como los drones.

1. INTRODUCCIÓN

1.1 Motivación

La agricultura tradicional se realiza siguiendo un patrón de homogeneidad, es decir, asume que todo el terreno del cultivo tiene características iguales, por tanto, no tiene en cuenta las propiedades cambiantes que se deben considerar a la hora de la producción agrícola y que retribuyen a la eficiencia y eficacia de la cadena de valor de la industria de este campo [13].

En ese sentido, la producción de los cultivos cambia dentro de los lotes en diversas zonas, por lo que habrá zonas que tengan una mayor productividad que otras y por ende no necesitarán la misma dosis de insumos (fungicidas, plaguicidas) y fertilizantes [14].

Estas propiedades cambiantes del suelo de cultivo pueden variar por factores pedogenéticos y acción antrópica [15].

Actualmente, la agricultura debe satisfacer el aumento de la población mundial y en este sentido demanda altas inversiones en capital humano, agua, fertilizantes y otros insumos para el manejo de los cultivos, que contribuye al deterioro ambiental y al deterioro de suelos y aguas [13], [16].

Igualmente, esta elevada demanda de alimentos agrícolas inclinó el desarrollo de la rentabilidad de los cultivos a través de métodos bióticos, los cuales, eran perjudiciales tanto para el entorno medioambiental como para el organismo humano [17].

No obstante, la demanda continúa creciendo y es totalmente necesario satisfacerla por cualquier método, en consecuencia, se inició a trabajar la agricultura de precisión desde el punto de vista de la caracterización del suelo y la fertilización por sitio específico [15].

La agricultura de precisión, desde esta perspectiva, se interpretó como el reconocimiento y entendimiento de la variabilidad espacial, y utilizar esta información para realizar un manejo diferencial que mejora la eficiencia y eficacia en la producción [13].

Otro concepto relacionado, es la sostenibilidad de la agricultura, la cual tiene como objetivo mantener la calidad del suelo, a partir de indicadores que permitan conocer si el uso y manejo de la tierra; es el indicado, con el fin de implementar cambios necesarios [14].

Así mismo, la agricultura de precisión se puede concebir como el desarrollo y la aplicación de tecnologías para establecer el comportamiento espacial de suelos y de esta forma optimizar los recursos agrícolas [16]

Esto se debe gracias a que en los últimos años el desarrollo de herramientas informáticas y de comunicación han permitido fortalecer y mejorar muchos procesos tanto en la parte industrial como en otros campos, con resultados muy eficientes [18].

Y, a la utilización de variedades mejoradas de maíz, trigo y otros granos; mediante el cultivo de una sola especie en un terreno durante todo el año, aplicación de grandes cantidades del agua, fertilizantes y plaguicidas, en general, al manejo de los factores bióticos, abióticos y sus interrelaciones [17].

Según la FAO (2013), un tercio de la población global deriva su sustento de la agricultura, y en economías emergentes esta puede representar hasta el 30 % del PIB [19].

En Colombia, el Plan Nacional de Desarrollo plantea como estrategia transversal la transformación del campo y crecimiento verde para fortalecer la competitividad del sector agrícola buscando modernizar procesos y proyectos.

Igualmente, define una política de crecimiento verde de largo plazo en la cual se definan los objetivos y metas de crecimiento económico sostenible. Dentro de sus estrategias se diseñará un programa de promoción de la investigación, desarrollo tecnológico e innovación para el fortalecimiento de la competitividad nacional y regional a partir de productos y actividades que contribuyan con el desarrollo sostenible y que aporten al crecimiento verde, dentro de estas actividades se encuentra la agricultura [20].

Por otro lado, el gobierno nacional considerando, la difícil panorámica en cuanto a innovación en el sector agropecuario, el Ministerio de Agricultura y Desarrollo Rural ha buscado dinamizar la inversión en investigación, desarrollo tecnológico e innovación, es así como en el período 2004 – 2008, se realizaron convocatorias públicas de cofinanciación con alianzas entre el sector productivo e investigador en demandas tecnológicas de las cadenas que sumaron $450 mil millones, 76% de los cuales corresponden a los dos últimos años [21].

Por tanto, dicho sector requiere la integración de diversos actores para alcanzar la modernización. En años recientes, empresas privadas y públicas pertenecientes al sector industrial, agrícola y de las TIC han unido esfuerzos para proyectar soluciones en el marco de la Agricultura de Precisión (AP), cuyo propósito es mejorar el rendimiento de cultivos, optimizar el uso de recursos, disminuir el impacto ambiental y facilitar la toma de decisiones estratégicas y económicas.

Por consiguiente, se hace necesario el desarrollo de un sistema de visión artificial de bajo costo para el control fitosanitario de cultivos de lulo que procese e identifique imágenes que evidencie la presencia de problemas fitosanitarios y que optimicen las condiciones del cultivo en pro de aumentar su desempeño y productividad.

En este sentido, se plantea un sistema que procese y segmente las imágenes obtenidas desde un vehículo aéreo no tripulado (UAS) y de esta forma pueda identificar la presencia de fitopatógenos en el cultivo.

El principal aporte de este trabajo radica en la utilización y aplicación avanzada de las nuevas tecnologías de visión por computador e inteligencia artificial, que se puedan desarrollar usando hardware y software de bajo coste [18].

Es decir, en la implementación de las tecnologías de la información y la comunicación (TIC) en los ciclos de vida de los cultivos [19].

Con una solución de este tipo se beneficiaría en primer lugar, el medio ambiente, ya que con la caracterización y monitoreo preciso del cultivo evitamos que se saturen algunas zonas del cultivo con insumos (fertilizantes, herbicidas, plaguicidas o fungicidas), que al no tener identificado los puntos de dosificación perjudica el ambiente, saturando y modificando muy bruscamente las propiedades del terreno de cultivo, contribuyendo al desgaste del suelo respecto a su fertilidad natural.

En segundo lugar, se beneficiarían los productores agrícolas al tener una herramienta eficiente de bajo coste que permita identificar de manera eficaz los fitopatógenos que se encuentren presentes en el cultivo, evitando así, el despilfarro de recursos tanto económicos que afecta la rentabilidad del negocio agrícola como de recursos naturales al maltratar el suelo y subsuelo del área de producción.

Adicionalmente, se contribuirá con nuevas aplicaciones en esta área, con el fin motivar a nuevos investigadores tanto de las instituciones de educación superior como al sector industrial a que sigan desarrollando prototipos y estudios que contribuyan a mejorar las condiciones de los productores agrícolas y de esta forma demostrar que las actividades agrícolas pueden ser muy rentables si se usa la tecnología.

1.2 Hipótesis.

Mediante el uso de la visión computacional y la inteligencia artificial, se pueden desarrollar sistemas de bajo de costo que identifican las señales de presencia de factores fitopatógenos en los cultivos de lulo.

1.3 Objetivos.

1.3.1 Objetivo General.

Diseñar un sistema de visión artificial de bajo costo para la detección de factores fitopatógenos en el cultivo del lulo.

1.3.2. Objetivos Específicos.

1.3.2.1. Identificar las librerías, funciones, filtros y algoritmos que más se nos ajusten a los requerimientos determinados para la aplicación.

1.3.2.2. Desarrollar el algoritmo que cumpla con todos los requerimientos y reportes, útiles para la ejecución o puesta en marcha del sistema.

1.3.2.3. Verificar el funcionamiento del sistema de visión artificial de bajo costo para la detección fitosanitaria en cultivos de lulo.

1.4. Metodología (Diseño Del Experimento)

La Investigación que se desarrollará en el presente trabajo será de tipo confirmatoria, ya que busca confirmar o desechar la hipótesis planteada inicialmente, tendrá un enfoque mixto, debido a que manejaremos variables tanto de tipología cuantitativa como cualitativa y el método de estudio será fenomenológico, ya que queremos comprender y analizar una situación dentro de un contexto.

Como se ve en la figura 1, esta investigación cumple una serie de etapas metodológicas secuenciales adaptadas de la norma Asme V&V 10.1-2012 (2012) para el proceso del desarrollo. En la primera fase se plantea, la visita al terreno de trabajo para recopilar información necesaria para identificar el fenómeno de estudio, conocer la morfología del terreno y tomar fotografías del objeto a analizar.

En la segunda fase se plantea la abstracción en donde se elabora el primer borrador conceptual del algoritmo y se identifican las características relevantes de interés del objeto de estudio.

En la tercera fase se procede a realizar un bosquejo o diagrama de cómo será el funcionamiento del código, donde se define los filtros a utilizar y los modelos matemáticos a aplicar en el procesamiento e identificación de las imágenes.

En la cuarta fase se define el código que más satisfaga los requerimientos.

Y en la quinta fase se realizan las verificaciones respectivas de aproximación y cálculo, para evaluar los resultados de simulación, y después ponerlo a prueba en el sitio de aplicación del prototipo o en un ambiente simulado y de esta forma entregar un sistema completamente evaluado [22].

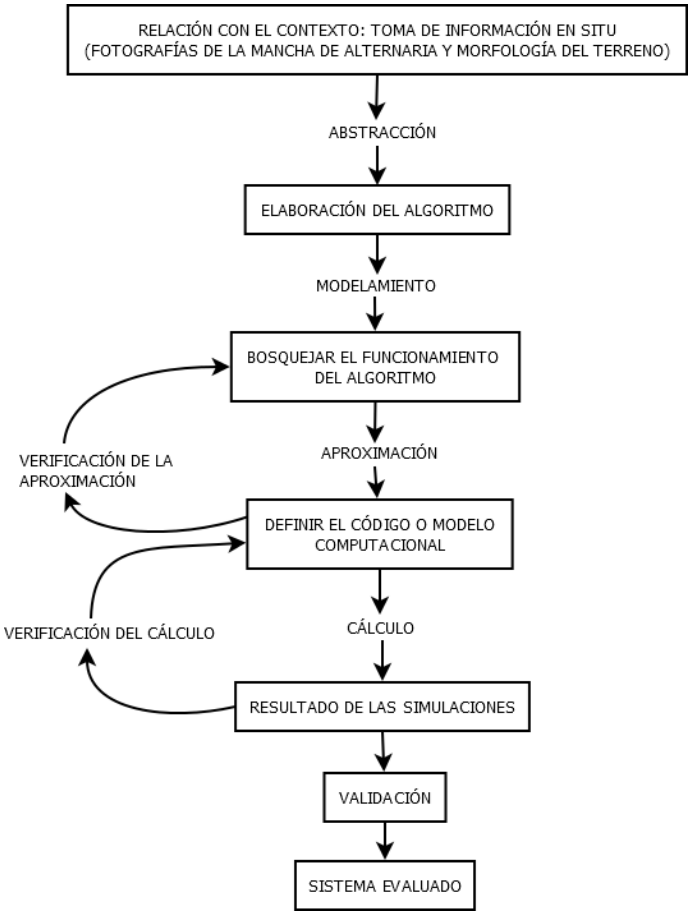

Figura 1. Metodología del trabajo de investigación

(Fuente: [22])

Así mismo, para el proceso de verificación, se aplicó las siguientes etapas secuenciales, donde se evaluó el funcionamiento del sistema y del algoritmo a través de un entorno simulado, y con los parámetros arrojados por la evaluación hecha en Matlab®.

En la primera etapa se evaluó el sistema de detección en un ambiente simulado, donde se ubicó en el plano una serie de fotografías de la enfermedad de estudio y se analizó su comportamiento respecto a dos variables de interés reportadas en varios textos de la literatura como es distancia del objeto al sensor de imagen y el porcentaje de detección en función del tiempo, posteriormente en la segunda fase se evaluó los algoritmos a través de un código implementado en Matlab ® en el cual, se obtendrá los parámetros de evaluación como F1, P y R.

En la figura 2, se observa el esquema de este proceso.

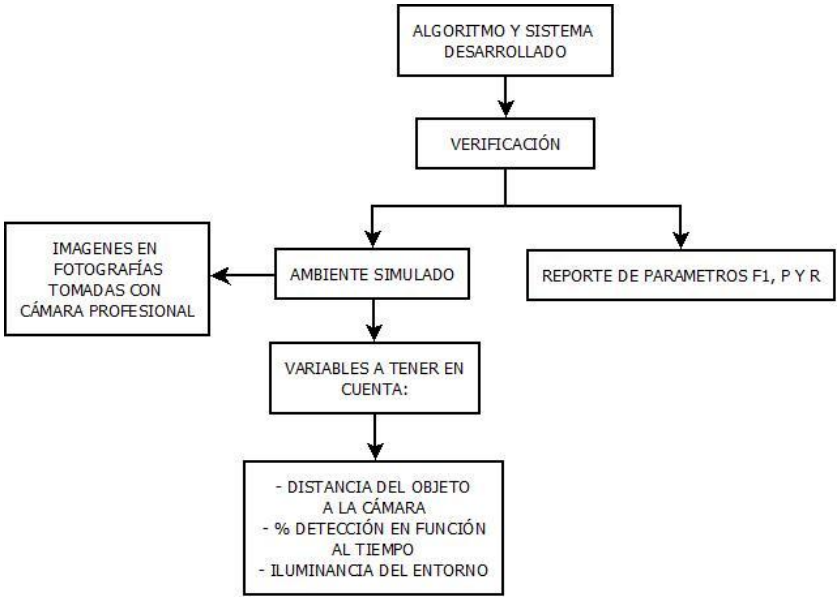

Figura 2. Proceso de verificación del sistema

(Fuente: Propia)

1.5. Organización de la memoria

El documento propuesto a continuación, esta conformado por cuatro capítulos organizados de la siguiente manera,en el primer capítulo de introducción, se argumentó el interés científico y técnico que tiene el autor para trabajar en la línea de instrumentación y sensórica, y técnicas de visión por computador o visión artificial y su combinación con la inteligencia artificial, presentándose cada uno de los objetivos a cumplir en el desarrollo de la investigación. En el segundo capítulo de marco teórico y antecedentes, se relacionan cada una de las bases teóricas necesarias para

poder llevar a cabo el desarrollo de la presente investigación, como la información general de los cultivos del lulo, explicándose las características principales de las múltiples plagas y fitopatógenos que se presentan en estos cultivos; se relacionan los diferentes algoritmos, librerias y funciones que se han desarrollado en el campo de la visión por computador y los softwares que se utilizan para estas aplicaciones como C, C++ y python. En el tercer capítulo, se presenta el desarrollo del sistema de detección, en esta sección encontraremos los hardware necesarios para desarrollar el sistema, al igual, que los equipos para la validación y toma de fotografías. Por otro lado, se enuncian los software requeridos para poder desarrollar el sistema de visión artificial, así mismo, los códigos base para generar el algoritmo de nuestra aplicación. En el cuarto capítulo, se efectúan las diferentes pruebas y verificaciones del algoritmo de visión artificial. Finalmente, en el quinto capítulo, se argumentan y clarifican cada una de las conclusiones obtenidas, y en el sexto capítulo se presentan las referencias bibliográficas analizadas.

En la figura 3, se presenta el diagrama de la estructura de la presente memoria.

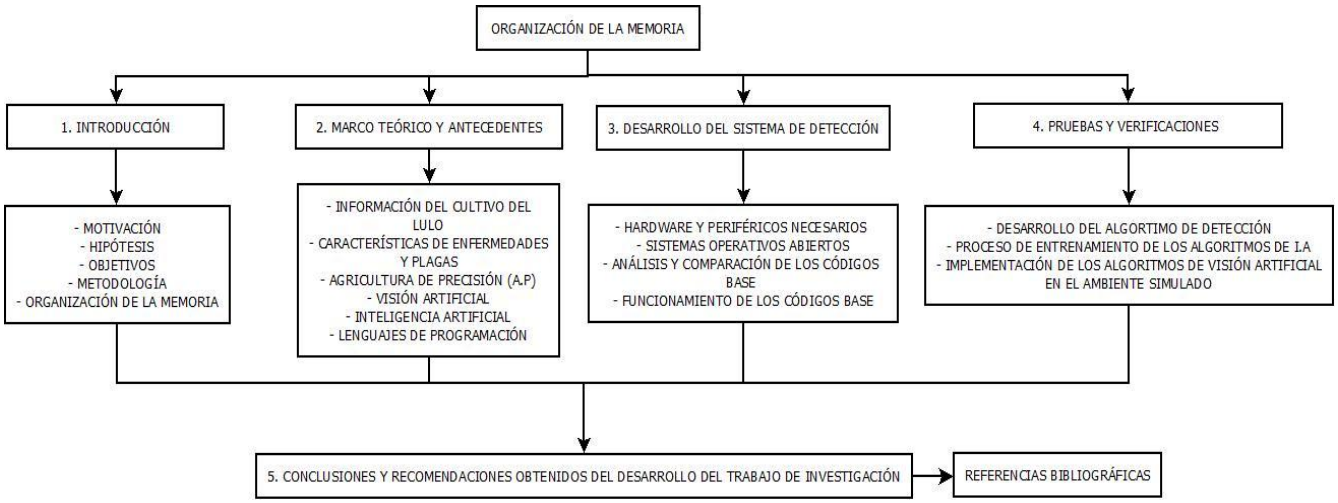

Figura 3. Estructura de la memoria

(Fuente: Propia)

2. MARCO TEÓRICO Y ANTECEDENTES

2.1. Introducción del Capítulo

En el siguiente capítulo se analizan cuatro temas importantes para el desarrollo del proyecto de investigación, por ende, este capítulo se dividirá en los siguientes títulos: Cultivo de lulo, factores fitopatógenos del lulo, agricultura de precisión y visión artificial, dentro del título del cultivo de lulo, se describió la importancia que tiene este cultivo en la economía y el sector social de la región, la clase y familia a la que pertenece este cultivo, las partes y demás conceptos de interés para el estudio de esta cultivo, en los factores fitopatógenos, se describió además de los fitopatógenos, las plagas y demás factores y elementos negativos que se pueden presentar en esta clase de cultivos, igualmente, se presentan imágenes de las patologías de las enfermedades identificadas hasta el momento por el ICA, donde es importante tener esta información, para la formulación de futuros proyectos relacionados, en el tema de agricultura de precisión, se relacionan los conceptos más destacados acerca de lo que es la agricultura de precisión, igualmente, las clases o tipologías de la A.P y las diferentes tecnologías usadas para el desarrollo de la misma, además se tomarán algunos trabajos destacados como ejemplos de la aplicación de tecnologías para implementar agricultura de precisión, relacionados al trabajo de investigación.

Por último, en el título de visión artificial se destaca las diferentes técnicas o metodologías usadas y los software que son útiles para el desarrollo de proyectos de visión artificial, así mismo, abordaremos temas relacionados a los diferentes filtros usados en la segmentación de las imágenes y a las diferentes fases de la visión artificial como son: preprocesamiento, segmentación, representación y descripción y; reconocimiento e interpretación, todo esto soportado desde su punto de vista teórico y práctico mostrando las diferentes librerías que realizan estas funciones, finalizando en la consolidación de los algoritmos más relacionados al trabajo y el desarrollo de uno que logre satisfacer las necesidades del proyecto, igualmente, mostraremos, las verificaciones y simulaciones que se realizarán como pruebas del sistema desarrollado.

2.2. Cultivo del lulo

El lulo es una especie de la familia de las solanáceas, ampliamente distribuida en la cordillera de los Andes que crece de manera espontánea en el sotobosque cerca de corrientes de agua. Su cultivo se adelanta principalmente en Perú, Ecuador, Colombia, Panamá, Costa Rica y Honduras. Esta especie produce una fruta con alta demanda en los mercados nacionales e internacionales dadas sus características y propiedades nutricionales. En Colombia se cultivan a plena exposición solar, las variedades: *S. quitoense quitoense* (sin espinas), *S. quitoense septentrional* (con espinas) y el híbrido "La Selva" obtenido a partir del cruzamiento del lulo de perro *(Solanum hirtum)* con el lulo de castilla *(Solanum quitoense)* [23].

En la figura 4 se muestra el fruto del cultivo del lulo.

Figura 4. Fruto del lulo

Fuente: Imagen tomada de http://observatorioruralbogota.gov.co/apc-aa-files/1e933e3a7f8b7d44268f40c145e93c17/DSC_3721.JPG)

En la figura 5, se muestra una toma panorámica de un cultivo de lulo en la región de Santo Domingo - Antioquia.

Figura 5. Cultivo de lulo en la región de Antioquia

(Fuente: Propia)

Igualmente, en la figura 6, se presenta las cantidades de producción de lulo por regiones en Colombia.

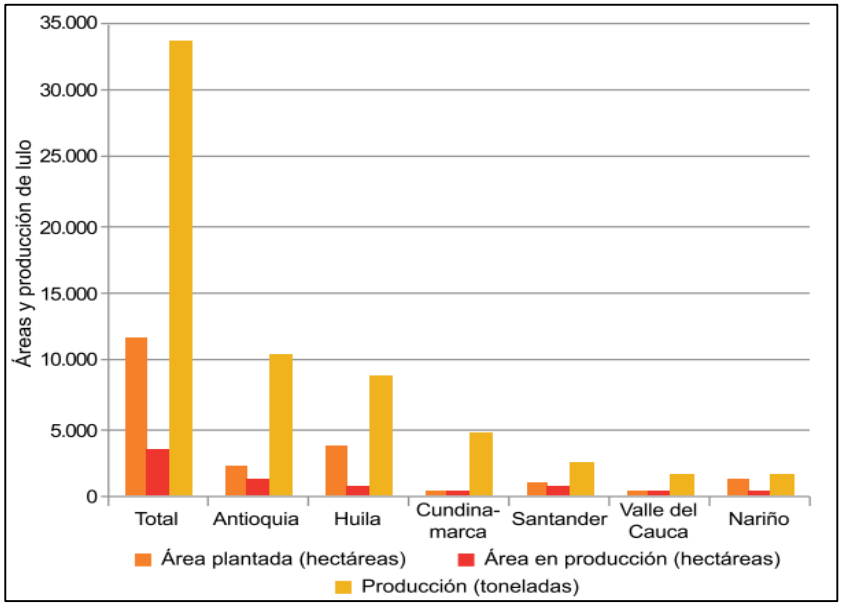

Figura 6. Producción de lulo en Colombia

(Fuente: DANE, ENA, 2015)

Como se puede observar en la figura 6, el departamento de Antioquia es el que más aporta en la producción del lulo.

2.2.1. Condiciones Agroecológicas

El cultivo del lulo se adapta y desarrolla muy bien en terrenos ubicados en alturas sobre el nivel del mar entre 1.300 a 2.200 m, con temperaturas de 14 a 18°C, precipitación anual de 1.500 a 2.200 milímetros de agua, humedad relativa o del ambiente alrededor del 80%, brillo solar de 4 a 6 horas diarias, pendiente menor del 40% y suelos, con; textura Franca – F, Franco arenosa – FA o Franco Arcillosa – FAr, moderadamente profundos de 50 a 75 centímetros, bien drenados y ligeramente ácidos con pH de 5.5 a 6.5 [4].

2.2.2. Afectaciones fisiológicas

Dependiendo de las condiciones medioambientales a las cuales se vea sometido el cultivo del lulo, se determina su expresión genotípica; un ambiente libre de condiciones de estrés contribuye al logro de altos rendimientos y mejores calidades en la producción.

Caso contrario, en condiciones de estrés medioambientales se favorece el desencadenamiento de una serie de trastornos fisiológicos en las plantas. Es así como, la absorción de agua y de nutrientes minerales necesarios para los procesos metabólicos, se ven interrumpidas por efectos de la sequía o del estrés hídrico e igualmente se da el cierre de las estomas, lo que impide el ingreso de CO_2 a la hoja y por tanto se interrumpe la fotosíntesis. Lo anterior puede generar: marchitez permanente; menor transporte de fotosintetizados; reducción del crecimiento radical, implicando menor absorción de agua y nutrientes minerales lo que afecta el desarrollo y crecimiento de órganos como yemas, flores y frutos. En resumen, La disfunción fisiológica causada por el estrés medioambiental severo o al ataque de insectos, precede y contribuye a la muerte de los árboles. De otra parte, bajo condiciones de sequía o déficit de humedad en el ambiente como en el suelo, es probable la ocurrencia de heladas que pueden llegar a quemar las flores, frutos recién cuajados, nuevos brotes y hojas, afectando drásticamente la producción [4].

Cabe indicar la importancia del agua como el elemento esencial de los tejidos vegetales, el cual corresponde al 80 y 90% del peso fresco, así mismo, interviene como disolvente, reactivo y en el mantenimiento de la turgencia de las plantas; lo anterior afirma la necesidad del suministro adecuado de agua en los cultivos de frutales durante las etapas de floración, cuajamiento y llenado de frutos, más si se ha observado que las hojas extrae agua de los frutos cuando se presenta estrés hídrico [4].

Por su parte, el adecuado suministro de nutrientes minerales asegura la expresión del potencial genético de las plantas, permitiendo el desarrollo óptimo del cultivo; en periodos de estrés hídrico la reducida disponibilidad de agua condiciona igualmente la disponibilidad de nutrientes, generando deficiencias nutricionales que afectan el proceso fisiológico y por tanto las etapas fenológicas del cultivo del lulo. En este sentido, se exponen los posibles daños que se presentan, por la carencia de algunos de los nutrientes como el nitrógeno, el fósforo, el azufre y el calcio [4].

2.3 Factores fitopatógenos del lulo

El lulo presenta un mapa fitosanitario bastante amplio, convirtiéndose en una de las especies más susceptibles al ataque de plagas y enfermedades. Entre los problemas considerados de mayor importancia en este cultivo se encuentran los marchitamientos tanto de origen bacteriano como fungoso, la gota o tizón en el follaje, la antracnosis en frutos, tallos y hojas, además de plagas como el pasador de los frutos. Otros factores que afectan negativamente la situación sanitaria del lulo son el manejo nutricional deficiente, las altas densidades de siembra y la deficiente calidad del material de propagación.

El manejo racional de las plagas, enfermedades y arvenses en cualquier cultivo requiere la adecuación de todas las labores agronómicas rutinarias al objetivo de la de sanidad de este; sin embargo, en muchos casos es necesario hacer intervenciones específicas, dada la agresividad e impacto de algunos de los problemas fitosanitarios. Dentro de un plan de manejo integrado, la eficacia de cada medida aumenta, reduciendo su impacto económico y ambiental.

Existen diferentes alternativas de manejo o intervención para cada una de las plagas, enfermedades y arvenses; varias de esas medidas, y específicamente el control químico, requieren la orientación y supervisión de un ingeniero agrónomo.

La inocuidad que debe ofrecer al consumidor la fruta fresca de lulo ha hecho que los ingenieros agrónomos que prestan asistencia técnica en este cultivo recomienden principalmente bioinsumos y plaguicidas de baja toxicidad. Por su parte, el ICA, mediante la Resolución 4754 de 2011, ha regulado la ampliación del uso de bioinsumos y plaguicidas químicos en cultivos menores [3].

2.3.1. Enfermedades

Las principales enfermedades del cultivo del lulo que son favorecidas por las condiciones de alta precipitación y humedad relativa son:

2.3.1.1. Tizón del lulo o gota (*Phytophthora infestans*)

Enfermedad causada por el protista con apariencia similar a un hongo: *Phytophthora infestans*. Se manifiesta en la parte aérea de la planta en forma de manchas que inicialmente son de color amarillo, pero poco a poco el tejido muere tomando un color oscuro. En casos severos, los tallos también presentan manchas necróticas irregulares que pueden causar la muerte de la planta. En el reverso de las manchas jóvenes se forma un crecimiento afelpado de color blanco, que está formado por numerosas estructuras del patógeno que diseminan la enfermedad a otras hojas o a otras plantas. Esta enfermedad afecta el cultivo desde el estado de plántula hasta el estado adulto y es prevalente en zonas y épocas lluviosas, alternadas con periodos de relativa sequía. Factores como la alta densidad de siembra y la deficiente poda de formación de la planta generan microclimas favorables, con alta humedad relativa. La enfermedad se presenta en zonas con temperatura entre 18 y 20°C. En la figura 7 se presenta la patología de esta enfermedad.

Figura 7. Phytophthora en lulo

(Fuente: [3])

2.3.1.2 Moho blanco, lama blanca, pudrición algodonosa

(Sclerotinia sclerotiorum) Es una enfermedad que afecta principalmente tallos, llegando a causar la muerte de toda la planta; es causada por el hongo *Sclerotinia sclerotiorum*. Este patógeno

generalmente se encuentra en el suelo y por eso la enfermedad empieza desde la parte inferior de la planta; las hojas se van marchitando y los tallos presentan manchas grandes e irregulares de color oscuro; el tejido afectado se cubre de una especie de algodón blanco formado por las estructuras del hongo. La enfermedad avanza rápidamente y puede afectar completamente la parte aérea, provocando muerte y momificación de frutos. Posteriormente se forman estructuras de supervivencia del hongo, las cuales son estructuras irregulares, endurecidas, de color negro, que se denominan esclerocios; éstas sobreviven en el suelo por largo tiempo. Esta enfermedad es favorecida por las lluvias continuas y las bajas temperaturas y afecta numerosos cultivos tales como habichuela, pepino, alverja, etc. En la figura 8 se ve una fotografía de esta patología.

Figura 8. Presencia en el tallo de moho blanco

(Fuente: [3])

2.3.1.3. Antracnosis del fruto

(*Colletotrichum gloesporioides*) Enfermedad favorecida por la alta humedad relativa y por las altas densidades de siembra. Se presenta en forma de manchas necróticas circulares, las cuales toman color negro y provocan el hundimiento del tejido, hasta cubrir todo el fruto, provocando su momificación y caída. En el centro de las manchas se desarrollan pequeñas estructuras de color rosado, las cuales corresponden al hongo causante: *Colletotrichum gloesporioides*. En ataques severos, suele presentarse también en tallos, pedúnculos y flores. La enfermedad se disemina

por el viento y el agua o mediante el contacto con insectos, animales o herramientas y afecta también otros cultivos como tomate de árbol, curuba, papaya, mango, etc.

En la figura 9 se muestra la presencia de esta enfermedad.

Figura 9. Antracnosis en tallo – Peñon (Cund)

(Fuente: I.A. Nelson Pachón [3])

2.3.1.4. Pudrición del tallo por Esclerotium

(*Sclerotium rolfsii*) Enfermedad que ocasiona volcamiento de plántulas en semillero y necrosis en el cuello de las plantas en campo. Se presenta causando lesiones necróticas únicamente en la base de la planta, las cuales provocan marchitamiento general y posteriormente su muerte. Es causada por el hongo *Sclerotium rolfsii*, el cual crece en las lesiones formando filamentos blancos llamados micelio; además produce estructuras de supervivencia en forma de pequeñas esferas duras de color café, conocidas como esclerocios, los cuales sobreviven en el suelo por largo tiempo (Hoyos, et al. 2008). La enfermedad afecta diferentes cultivos y se presenta en zonas lluviosas y suelos livianos [3].

En la figura 10, se evidencia la presencia de esta enfermedad.

Figura 10. Pudrición del tallo por Esclerotium

(Fuente: I.A. Nelson Pachón [3])

2.3.1.5 Marchitez vascular (Fusarium oxysporum)

El hongo causante inicia la infección en el sistema radical, penetrando por las lesiones causadas por las herramientas o los nematodos. Si se hace un corte transversal en el cuello de la raíz y el tallo, se observa una coloración rosada o morada oscura en los haces vasculares; este síntoma se manifiesta igualmente en raíces, ramas superiores y pecíolos de las hojas. Las plantas muestran flacidez y clorosis ascendentes en la medida en que el hongo *Fusarium oxysporum* invade los tejidos vasculares, hasta llegar a su muerte. Los cultivos con deficiente nutrición son más susceptibles a la enfermedad. El patógeno es específico del lulo, pero puede sobrevivir largo tiempo en el suelo [3].

En la figura 11, se muestra el ataque de este patógeno

Figura 11.Corte transversal de tallo, mancha café evidencia de fusarium

(Fuente: [3])

2.3.1.6. Agalla radical (Meloidogyne spp)

Los nematodos son parásitos que infectan la raíz de la planta formando nudos que impiden la translocación de agua y nutrientes; por esta razón, los síntomas en la parte aérea son clorosis, retraso en el desarrollo, marchitez de las hojas y en general, menor tamaño de la planta. El síntoma local más característico es la formación de nudos o agallas en las raíces. Por su forma de alimentación con ayuda de un estilete, los nematodos facilitan la penetración de bacterias y hongos, aumentando por ello la susceptibilidad de las plantas a la pudrición de raíces. Los nematodos del género *Meloidogyne* afectan numerosas especies vegetales, por lo que una vez llegan a un terreno, su población va en aumento con la siembra sucesiva de plantas susceptibles [3].

En la figura 12, se muestra la fotografía donde se evidencia los síntomas de esta enfermedad.

Figura 12. Amarillamiento, síntoma de presencia nemátodos

(Fuente: [3])

Y por último mencionaremos la enfermedad que vamos a usar para la realización de nuestro trabajo de investigación, la cual es la mancha de Alternaria.

2.3.1.7. Alternaria

La Alternaria se caracteriza porque en las zonas atacadas aparecen unas manchas de color negro o pardas («negrón»), bien delimitadas, que en algunos casos pueden estar rodeadas por una o varias aureolas concéntricas.

Los tratamientos deberán ser periódicos y preventivos cada 10-15 días con fungicidas, especialmente si en otros años ha aparecido. Sirven, por ejemplo, los clásicos Zineb, Maneb, Mancozeb, etc., Cobre, Benzimidazol [24].

En la figura 13 se presenta la patología de mancha de alternaría rescatada de una referencia bibliográfica.

Figura 13. Patología de Mancha de Alternaria

(Fuente: [23])

En la figura 14, se muestra la imagen de la patología descrita anteriormente, pero en la finca que nos permitió tomar los datos para el desarrollo de la investigación, esta se encuentra ubicada en el municipio de Santo Domingo, subregión Nordeste del departamento de Antioquia, a 1975 m sobre el nivel del mar, finca La Corraleja.

Figura 14. Mancha de Alternaria en la Finca “La Corraleja”

(Fuente: Autor)

2.4. Agricultura de Precisión (A.P)

En los últimos años la agricultura de precisión ha ganado una gran importancia en la comunidad agrícola, pero a pesar de la gran mayoría de avances y desarrollos en el área muy pocos agricultores practican este concepto.

Se pueden describir dos líneas de desarrollo de la agricultura de precisión para entender mejor sus antecedentes y su evolución hasta el día de hoy: la agronómica y la ingeniería agrícola.

La línea agronómica, refiere a los trabajos de las décadas del ‘70 y ‘80, en Minnesota, Estados Unidos, sobre la utilización de métodos de investigación de campo para conocer mejor la

variabilidad de los factores de suelo y planta, incluyendo análisis de suelo, muestreo del suelo, fotografía del área y análisis de cultivos. Gracias a la unión de esfuerzos de las empresas CENEX, FARMERS Union Central Exchange Inc. y la compañía de computadoras Control Data Corporation, ambas con sede en Saint Paul y Miniápolis, Minnesota, Estados Unidos (EE.UU.), fue posible establecer el primer concepto de variabilidad de suelo y planta en los campos, así como los potenciales beneficios de su gerenciamiento por zonas de manejo, en vez de toda el área sembrada [25].

La línea de la ingeniería agrícola se refiere a la evolución de las máquinas agrícolas, utilizando sensores y sistemas de posicionamiento global (GPS) para mapeo y aplicación de insumos con dosis variada. La empresa Massey Ferguson fue, en 1982, la primera compañía en producir una cosechadora comercial con sistema de mapeo de productividad de granos. Luego, ya al comienzo de los años ‘90, aparecieron los proyectos de John Deere, Case, AGCO y New Holland.

En la mayoría de las cosechadoras disponibles en el mercado mundial, que operan dentro del concepto de agricultura de precisión, se encuentran los siguientes componentes del sistema de monitoreo de rendimiento de granos [26]:

- Sensores para medir el flujo de los granos
- Humedad de los granos
- Velocidad de cosecha
- Indicador de posición de la plataforma de corte de la cosechadora
- Monitor de funciones de las operaciones
- GPS.

Los mapas de rendimiento de granos se elaboran a partir de la información recibida por esos sensores y procesada por un software como, por ejemplo, un Sistema de Información Geográfica (SIG). A lo largo del tiempo, se ha producido el siguiente esquema de desarrollo:

- 1982 desarrollo del monitor de rendimiento
- 1984 desarrollo del sistema de posicionamiento
- 1985/96 Desarrollo de la tecnología de dosis variable
- 1991 primer sistema de monitoreo de rendimiento vendido en Europa
- 1996 Lanzamiento de los modelos comerciales de monitoreo de rendimiento: Field Star, Green Star, etc [26]

Así mismo, la gran cantidad de técnicas que se están trabajando en la actualidad para la implementación de la agricultura de precisión son innumerables, a continuación, relacionamos una serie de trabajos y estudios que se han venido desarrollando en este tema:

Faiçal y otros, (2017) propone un sistema informático que es capaz de adaptar de forma autónoma el control UAV mientras se mantiene la deposición precisa de pesticidas en los campos objetivo. En este estudio son evaluadas diferentes versiones de la propuesta, con metaheurísticas de adaptación de rutas autónomas basadas en Algoritmos Genéticos, Optimización de enjambre de Partículas, *Simulated Annealing* y *Hill-Climbing* con el fin de optimizar la intensidad de los cambios de ruta. Adicionalmente, este estudio evalúa el uso de una estación de control hardware incorporado para ejecutar la metaheurística de adaptación de la ruta. Los resultados experimentales mostraron que el sistema basado en computadoras con metaheurísticas de cambio de rutas autónomas proporciona cambios precisos en la ruta de vuelo del UAV, con una deposición más precisa del pesticida y menos daños [27].

En este sentido, Coates, Delwiche, Broad, y Holler, (2013) exponen un sistema de accionamiento de válvula incluido el desarrollo de firmware de nodo personalizado, hardware y firmware del actuador, en este proyecto se instalaron treinta y cuatro actuadores de válvulas implementados en el campo para controlar 54 válvulas y monitorear 6 contadores de agua [28].

Por otro lado, Gimenez, Herrera, Tosetti, y Carelli, (2015) plantean una técnica para la cartografía de una arboleda frutal por un robot móvil, que utiliza sólo información láser frontal del entorno y la posición exacta de las esquinas de la arboleda. Este método se basa en la resolución de un problema de optimización con restricciones no lineales, lo que reduce los errores inherentes al proceso de medición, garantizando una construcción de mapas eficiente y precisa. El algoritmo resultante se probó en un entorno de huerto real. Para ello, se desarrolló también un método de filtrado de datos capaz de cumplir eficientemente la correspondencia observación-característica. El error medio máximo obtenido por la metodología en las simulaciones fue de unos 13 cm, y en la experimentación real fue de aproximadamente 36 cm [29].

Así mismo, Torres-Sánnchez, Peña, de Castro, y López-Granados, (2014) en su artículo, utilizó un UAV equipado con una cámara comercial (espectro visible) para la adquisición de imágenes de ultra alta resolución en un campo de trigo en el período de la primera temporada. A partir de estas imágenes, se calcularon y evaluaron seis índices espectrales visibles (CIVE, ExG, ExGR, Woebbecke Index, NGRDI, VEG) y dos combinaciones de estos índices para el mapeo de

fracción de vegetación, para estudiar la influencia de la altitud de vuelo (30 y 60 m) en los días después de la siembra (DAS) de 35 a 75 DAS sobre la exactitud de clasificación. Los índices ExG y VEG obtuvieron la mejor precisión en la cartografía de la fracción de vegetación, con valores que oscilaron entre 87.73% y 91.99% a una altitud de 30mflight y de 83.74% a 87.82% a una altitud de vuelo de 60m. Estos índices también fueron espacial y temporalmente consistentes, permitiendo un mapeo preciso de la vegetación en todo el campo de trigo en cualquier fecha. Esto proporciona evidencia de que los índices espectrales visibles derivados de imágenes adquiridas usando una cámara de bajo costo a bordo de un UAV que vuela a bajas altitudes son una herramienta adecuada para usar y para discriminar la vegetación en los campos de trigo en la estación temprana. Este trabajo permitió concluir que la utilización de esta tecnología en aplicaciones de agricultura de precisión tales como el manejo temprano de malezas específicas en sitio, son esenciales para la clasificación de malezas en diversos cultivos [6].

Igualmente, Mohapatra y Lenka, (2016) se centra en dos estrategias de optimización, por ejemplo, Gradiente Escalado Conjugado y BFGS Quasi-Newton basado en los algoritmos de red neuronal utilizados para predecir la necesidad horaria de suelo. Los rendimientos de predicción de estas dos técnicas de optimización también se estudiaron mediante el cálculo de MSE (error cuadrático medio), RMSE (error cuadrático medio) y error R cuadrado. Los cálculos se usan para la predicción del MC del suelo en cada avance de una hora considerando once parámetros distintivos del suelo y del medio ambiente. La mejor técnica se utilizó para la predicción final. El sistema propuesto es un sistema híbrido utilizado para resolver un solo problema que es la generación de mejores sugerencias de riego para los agricultores[30].

Doering y otros, (2016) Presentan el modelo, diseño y la caracterización de un sistema de cámaras multiespectral que es implementado en un UAV, a partir de dos cámaras comunes que son totalmente compatibles con hardware de fuente libre comerciales de bajo costo, como las tarjetas Raspberry Pi, que se utilizan para recopilar, procesar y almacenar información [7].

Así mismo, Rasmussen y otros, (2016) evaluó si los Vis (índices de vegetación) derivados de las cámaras de grado de consumo montado en Vehículos aéreos no tripulados o UAV por sus iniciales en inglés son fiables y si hay alguna deficiencia en la adquisición de imágenes y el análisis, que deben abordarse antes de su aplicación general. Este objetivo fue investigado a partir de cámaras de tipo UAV, CVA (True Color) y CIR (color-infrarrojo), cuatro VIs diferentes (ExG, NGRDI, NDVI y ENDVI), con altitudes en el rango de 30-100 m, en diferentes condiciones de iluminación ambiental y dos paquetes de software diferentes para procesar imágenes. Los

resultados se compararon con grabaciones en tierra por cámaras de consumo y sensores multiespectrales. Se realizaron experimentos de campo en cereales para evaluar el sistema. El estudio mostró que los VIs basados en imágenes de UAV tienen la misma capacidad de cuantificar las respuestas de los cultivos a los tratamientos experimentales como las grabaciones terrestres con cámaras y sensores avanzados. Sin embargo, hay deficiencias que deben tenerse en cuenta: (1) la variación angular en la reflectancia (reflectancia bidireccional), (2) la costura y (3) las fluctuaciones de la luz ambiente. La reflectancia bidireccional era tan extensa que podía conducir a conclusiones engañosas en condiciones soleadas y este efecto podría amplificarse aún más por la costura. Se demuestra un procedimiento para evitar los impactos de la reflectancia bidireccional cuando se recortan tramas de imágenes individuales y se sugiere un procedimiento para corregir estas imágenes. La cámara, los VI y la altitud de adquisición de la imagen fueron de menor importancia, pero las condiciones fluctuantes de iluminación ambiental son un tema que debe ser abordado en futuros estudios [31].

Entre tanto, Ribeiro-Gomes, Hernandez-Lopez, Ballesteros, y Moreno, (2016) desarrollaron una metodología para reducir el costo de generar productos geomáticos a través de: 1) detección automática de imágenes borrosas en un conjunto de imágenes que fue capturado con un UAV mediante el establecimiento de un indicador numérico que describe el nivel de desenfoque en las imágenes y 2) eliminando la necesidad de medir puntos de control de tierra (GCP) para georreferenciar los productos geomáticos finales con la orientación exterior aproximada de las imágenes y puntos de control de los productos geomáticos existentes. El tiempo que se ahorró en la realización de las tareas manuales que se requerían para generar productos geomáticos, que son las tareas que influyen fuertemente en el costo, se redujo en nuestro estudio de 40 ha por 65 a 69%, lo que corresponde a un ahorro de costos de € 200 a 225 [32].

Por otro lado, J. Torres-Sánchez, F. López-Granados, y J. M. Peña, (2015) desarrollaron un innovador algoritmo OBIA de umbral basado en el método de Otsu y estudia cómo los resultados de este algoritmo se ven afectados por los diferentes parámetros de segmentación (escala, forma y compacidad). Junto con la descripción general del procedimiento, se aplicó específicamente para la detección de vegetación en imágenes de captación remota capturadas con dos sensores, (una cámara visible convencional y una cámara multiespectral) montada sobre un vehículo aéreo no tripulado (UAV) y adquirida en campos de tres diferentes cultivos herbáceos (maíz, girasol y trigo). Las pruebas analizaron el desempeño del algoritmo OBIA para clasificar la cobertura vegetal afectada por diferentes umbrales automáticamente seleccionados, calculados en las imágenes a partir de dos índices de vegetación: el Exceso de Verde (ExG) y el Índice de

Vegetación de Diferencia Normalizada (NDVI). El parámetro escala de segmentación afectó los histogramas del índice de vegetación, lo que condujo a cambios en la estimación automática del valor umbral óptimo para los índices de vegetación. Los otros parámetros implicados en el procedimiento de segmentación (es decir, forma y compacidad) mostraron una influencia menor en la precisión de clasificación. Aumentando el tamaño del objeto, el error de clasificación disminuyó hasta que se alcanzó un óptimo. Después de este valor óptimo, el aumento del tamaño del objeto produjo errores mayores.

Zarco-Tejada, Diaz-Varela, Angileri, y Loudjani, (2014) en su estudio proporciona información sobre la evaluación de la recuperación de parámetros biofísicos del dosel utilizando sensores pasivos y específicamente en la cuantificación de la altura del árbol en un dosel discontinuo usando una cámara de bajo costo a bordo de un vehículo aéreo no tripulado. En este estudio, la configuración del UAV eléctrico que transportaba la carga útil de la cámara permitió la adquisición de 158 ha en un solo vuelo. El sistema de cámara hizo posible la adquisición de imágenes de alta resolución (VHR) (5 cm pixel-1) para generar orto-mosaicos y modelos de superficie digital (DSMs) a través de métodos automáticos de reconstrucción 3D. El UAV siguió planos de vuelo previamente diseñados sobre cada sitio de estudio para asegurar la adquisición de la imagen con grandes superposiciones a lo largo ya lo largo (es decir, más del 80%) usando una cuadrícula de líneas de vuelo paralelas y perpendiculares. El método de validación consistió en tomar medidas de campo de la altura de un total de 152 árboles en dos áreas de estudio diferentes utilizando un GPS en modo cinemático en tiempo real (RTK). Los resultados de la evaluación de validación realizada para estimar la altura de los árboles a partir de los DSM de VHR arrojaron $R^2 = 0.83$, un error cuadrático medio total (RMSE) de 35 cm y un error cuadrático medio relativo (R-RMSE) del 11,5% para los árboles con alturas entre 1,16 y 4,38 m. Una evaluación realizada sobre los efectos de la resolución espacial de las imágenes de entrada adquiridas por el UAV, sobre el método de reconstrucción y generación DSM, demostró, relaciones estables para resoluciones de píxeles entre 5 y 30 cm que se degradaron rápidamente para las imágenes de entrada con resoluciones de píxeles más bajas de 35 cm. Los valores RMSE y R-RMSE obtenidos en función de la resolución de píxeles de entrada mostraron errores en la cuantificación de árboles por debajo del 15% cuando 30 cm de resolución de píxeles 1 de imágenes se utilizó para generar el DSMs. El estudio realizado en dos huertos con este sistema de UAV y el método de foto-reconstrucción resaltó que un enfoque barato basado en cámaras de consumo a bordo de una plataforma aérea no tripulada lanzada manualmente puede proporcionar precisiones

comparables a las de la costosa y computacionalmente más compleja luz detección y distribución (LIDAR) actualmente operados para aplicaciones agrícolas y ambientales [33].

Por otro lado, Gago y otros, (2015) revisaron aplicaciones de diversos tipos de UAV usando diferentes sensores remotos y compararon su desempeño con datos de la planta de tierra-verdad. Varios índices de reflectancia, tales como NDVI, TCARI / OSAVI y PRInorm obtenidos de UAV han mostraron correlaciones positivas relacionadas con indicadores de estrés hídrico como el potencial hídrico y la conductancia estomática. Sin embargo, se han comportado de manera diferente en diversos cultivos; por tanto, sus usos y aplicaciones también se discuten en el estudio. Las imágenes térmicas son también una tecnología común de teledetección utilizada para evaluar el estrés hídrico en las plantas, a través de índices térmicos (calculados utilizando superficies artificiales como referencias), estimaciones de la diferencia entre la temperatura de la copa y el aire. Estos índices han mostrado un gran potencial para determinar la heterogeneidad del esfuerzo de campo utilizando plataformas aéreas no tripuladas. También se ha propuesto que la fluorescencia de la clorofila podría ser un indicador aún mejor de la fotosíntesis de las plantas y la eficiencia del uso del agua bajo estrés hídrico [34].

Santesteban y otros, (2017) evaluaron hasta qué punto la imagen térmica de alta resolución permite evaluar la variabilidad instantánea y estacional del estado del agua dentro de un viñedo. La novedad y la importancia de su enfoque es que el vehículo aéreo no tripulado (UAV) específicamente diseñado y construido proporcionó imágenes de muy alta resolución (píxeles <9 cm) y que fue utilizado en una superficie comercialmente relevante (7,5 ha). Esta configuración se utilizó para obtener Crop Water Stress Index (CWSI) a partir de imágenes térmicas en un día de cielo despejado. Los valores de CWSI fueron comparados con el potencial de agua del tallo y la conductancia estomática medida en 14 sitios de muestreo en el viñedo en el momento en que se adquirieron las imágenes. Con el fin de evaluar el potencial de CWSI adquiridos en un solo día para estimar dentro de vides patrones de variación en el estado del agua, con el modelado espacial que se utilizó [35].

Senthilnath y otros, (2016) presentaron la clasificación espectral-espacial de imágenes RGB de alta resolución espacial obtenidas de vehículos aéreos no tripulados (UAV) para la detección de tomates en la imagen. Se utilizó el criterio de información bayesiano (BIC) para determinar el número óptimo de racimos para la imagen. La agrupación espectral se llevó a cabo utilizando algoritmos de K-means, maximización de expectativas (EM) y autoorganización (SOM) para categorizar los píxeles en dos grupos, es decir, tomates y no tomates. Debido a la similitud en las

intensidades espectrales, algunos de los píxeles no tomate fueron agrupados en el grupo de tomate y con el fin de eliminarlos, la segmentación espacial se realizó en la imagen. La segmentación espacial se llevó a cabo utilizando operaciones morfológicas y estableciendo umbrales para propiedades geométricas. El número de píxeles agrupados en el clúster de tomate es diferente para cada método de agrupación. EM no recoge los parches de tierra como píxeles de tomate. Como resultado, el tamaño de los tomates recogidos es diferente de K-means y SOM. Dado que los valores de umbral elegidos para llevar a cabo la segmentación espacial son dependientes de la forma y el tamaño, diferentes valores de umbral son aplicado a diferentes métodos de agrupación en este trabajo se utilizaron dos imágenes representativas de UAV capturadas a diferentes alturas del suelo para demostrar el rendimiento del método propuesto [9].

Por otro lado, Polo, Hornero, Duijneveld, García, y Casas, (2015) tambien proponen un sistema de servidor de monitoreo de ambiente agrícola utilizando una Red de Sensores Inalámbricos (WSN) de bajo costo. Varios nodos de sensores están dispersos en campos de varios kilómetros de tamaño, y proponemos la recopilación de la información almacenada en los nodos por un nodo móvil, o mula. Para cubrir largas distancias en un corto período de tiempo, se utiliza un vehículo aéreo no tripulado (UAV), que recupera los datos almacenados en los nodos de tierra. Además, el UAV puede ser utilizado para adquirir información adicional y para realizar acciones. Su posición elevada permite observar el campo con una perspectiva que es útil para detectar cambios que afectan a los cultivos, tales como plagas, enfermedades, cambios significativos en la humedad del suelo, sequía o inundación [36].

Para destacar y dar un análisis concreto de estas investigaciones, en la tabla 1, se hace una comparación de los trabajos más relevantes en esta área.

Tabla 1. Cuadro comparativo de trabajos relevantes al área de estudio

Título del trabajo	Autores	Año	Hardware y software de bajo Costo	Detección de enfermedades en el cultivo	Desarrollo de Estrategia de Visión e inteligencia artificial	Procesamiento en Tiempo Real
Multi-temporal mapping of the vegetation fraction in early-season wheat fields using images from UAV	Torres-Sánnchez, Peña, de Castro, & López-Granados, (2014)	2014	SI	NO	NO APLICA (Agisoft PhotoScan Professional Edition)	NO
Neural Network Pattern Classification and Weather Dependent Fuzzy Logic Model for Irrigation Control in WSN Based Precision Agriculture	Mohapatra & Lenka, (2016)	2016	NO	NO	Redes Neuronales Lógica Difusa Gradiente Escalado Conjugado y BFGS Quasi-Newton	SI
MDE-based Development of a Multispectral Camera for Precision Agriculture	Doering et al., (2016)	2016	SI	NO	NO APLICA	NO
Are vegetation indices derived from consumer-grade cameras mounted on UAVs sufficiently reliable for assessing experimental plots?	Rasmussen et al., (2016)	2016	NO	NO	K-means y algoritmo EM	NO
An automatic object-based method for optimal thresholding in UAV images: Application for vegetation detection in herbaceous crops	Torres-Sánchez, López-Granados, & Peña, (2015)	2015	NO	NO	OBIA de Umbral basado en el método de Otsu	NO

Tree height quantification using very high resolution imagery acquired from an unmanned aerial vehicle (UAV) and automatic 3D photo-reconstruction methods	Zarco-Tejada, Diaz-Varela, Angileri, & Loudjani, (2014)	2014	SI	NO	NO APLICA (pix4UAV)	NO
Detection of tomatoes using spectral-spatial methods in remotely sensed RGB images captured by UAV	Senthilnath et al., (2016)	2016	SI	NO	K-means Maximización de expectativas (EM) y autoorganización (SOM)	NO
Controlador borroso multivariable para el ajuste de tratamientos en agricultura de precisión	Burgos-Artizzu, Ribeiro, & Santos, (2007)	2007	NO	NO	Segmentación de Imagen por índice de color	NO
Classification of imbalanced ECG beats using re-sampling techniques and AdaBoost ensemble classifie	Rajesh & Dhuli, (2018)	2018	NO	NO	Adaboost, Boosting	NO

(Fuente: Propia)

Con la ayuda de la tabla 1, se pueden identificar los siguientes hallazgos de los sistemas encontrados: algunos de los trabajos implementan hardware y software libre, no obstante, todavía en alguno de ellos se cuentan con cámaras y software que aún son costos. Todos los trabajos relacionados procesan imágenes para identificar propiedades del terreno o del cultivo, o para identificar malezas o plagas, ninguno de los trabajos estudiados se ha desarrollado para identificar patologías de hongos en el mismo cultivo. Los trabajos descritos anteriormente cuentan con una etapa off-line de procesamiento de las imágenes, por lo que se deben descargar de la cámara y posteriormente llevarlas a un ordenador para procesarlas y aplicarles el algoritmo de segmentación correspondiente.

En algunos de los casos se utilizan cámaras multiespectrales, que de cierta forma facilita la segmentación a través de filtros usados posteriormente, en muy pocos casos se ve el uso de cámaras de bajo costo y de espectro visible.

Igualmente, con la información obtenida de la literatura referente a los sistemas automáticos de AP, que aplican diferentes técnicas o estrategias de control, que van desde el control analógico tradicional hasta el procesamiento de imágenes a través de métodos de inteligencia artificial, se pudo, realizar la síntesis del diagrama que se ve en la figura 15, el cual menciona o ilustra los diferentes medios de captación de las variables de entrada, las diferentes estrategias de control aplicadas y las técnicas de actuación implementadas en situ en el cultivo.

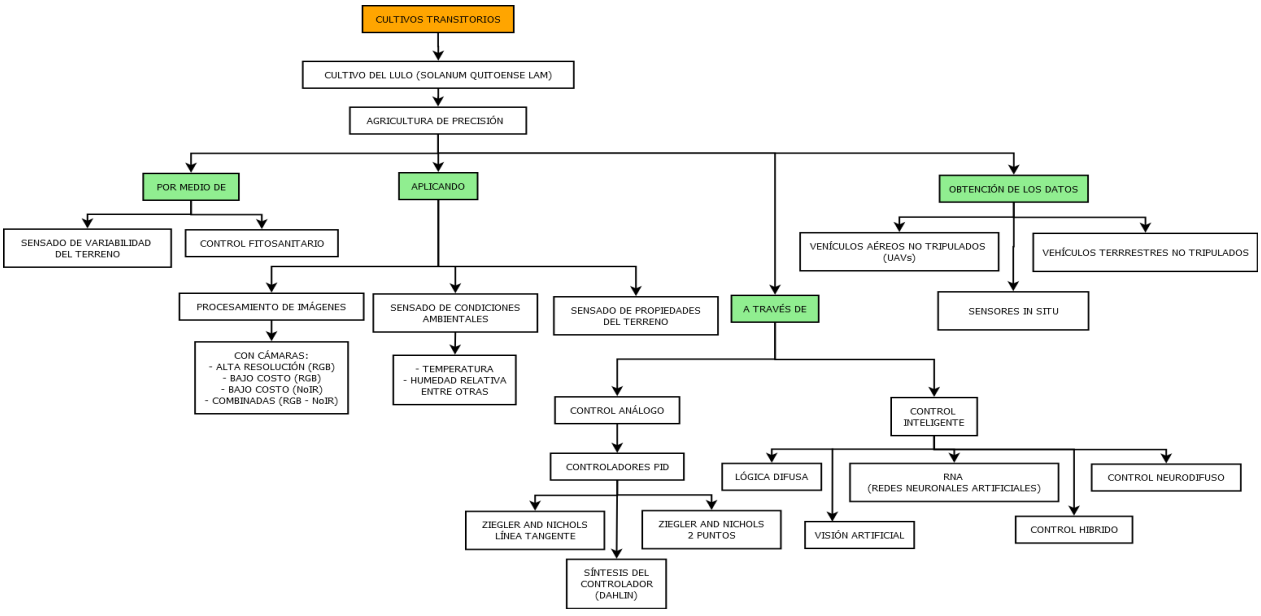

Figura 15. Técnicas de Agricultura de Precisión (A.P)

(Fuente: Propia)

2.4.1 Optimización del uso de insumos a través de la agricultura de precisión

Los sistemas tradicionales de producción tratan a las propiedades agrícolas de forma homogénea, tomando como base las condiciones promedio de las extensas áreas de producción, para implementar las acciones correctivas de los factores limitantes.

Con el fin de obtener unos sistemas de producción más competitivos y aumentar la eficiencia agronómica del sector productivo, se incorporaron nuevas técnicas para incrementar y/o mantener la productividad de los cultivos, buscando, al mismo tiempo, reducir los costos de producción. En ese contexto, la optimización del uso de insumos a través de la agricultura de precisión es una alternativa que establece una manera diferenciada de manejo del sistema de producción, buscando promover la estabilidad de la producción a través de la maximización del retorno económico y preservando el medio ambiente.

Conceptualmente, la agricultura de precisión es una nueva forma integrada de gerenciamiento de la información de los cultivos, basada en la existencia de la variabilidad espacial y temporal de la unidad mínima de manejo en la agricultura tradicional [26].

La adopción de la agricultura de precisión, no solamente como utilización de tecnologías de información, sino como concepto, es un potencial para la racionalización del sistema de producción agrícola moderno como consecuencia de:

- Optimización de la cantidad de agroquímicos aplicados en los suelos y cultivos
- Consecuente reducción de los costos de producción y de la contaminación ambiental
- Mejora de la calidad de las cosechas.

Igualmente, en los últimos años se ha producido un importante avance con la aplicación de sistemas de control y automatización en agricultura apareciendo el concepto de Agricultura de Precisión (AP) que engloba un conjunto de técnicas de cultivo dirigidas a ajustar el uso de agroquímicos considerando la diversidad tanto del medio físico como del biológico. Lo que se traduce en una reducción de los costes de producción y una gestión agrícola más respetuosa con el medioambiente [38] .

Si bien es un tema de investigación relativamente nuevo, se han logrado muchos avances, principalmente en el desarrollo de máquinas e implementos que permiten el manejo localizado en base a mapas. Los recursos más avanzados en tecnología de información hoy disponibles, como

los sistemas de posicionamiento global (GPS), los (SIG), los sistemas de control y adquisición de datos, sensores y actuadores, entre otros, están cada vez más presentes en el campo.

Adicionalmente, la aplicación del control tradicional (Controladores PI y PID) para la automatización de sistemas de irrigación por canal, que optimiza en gran medida el recurso hídrico, en los cultivos de gran área [40], [41]

Así mismo, la implementación de estrategias de control de inteligencia artificial, ha permitido el desarrollo de controladores capaz de manejar procesos complejos (plantas no lineales de orden elevado, con parámetros internos que varían en el tiempo, dependientes del entorno, etc) [42], en este sentido, según lo plantea Santos (2011), aplica los procesos de la AP.

A pesar de ese avance tecnológico, Cano Marchal, Gómez Ortega, Aguilera Puerto, & Gámez García (2011), evidencian que en España, la automatización y control de la producción agrícola del aceite de oliva en las almazaras es de un nivel medio y, que el grado de implementación depende en gran medida al tamaño de la empresa como se puede observar en la figura 16 [44] .

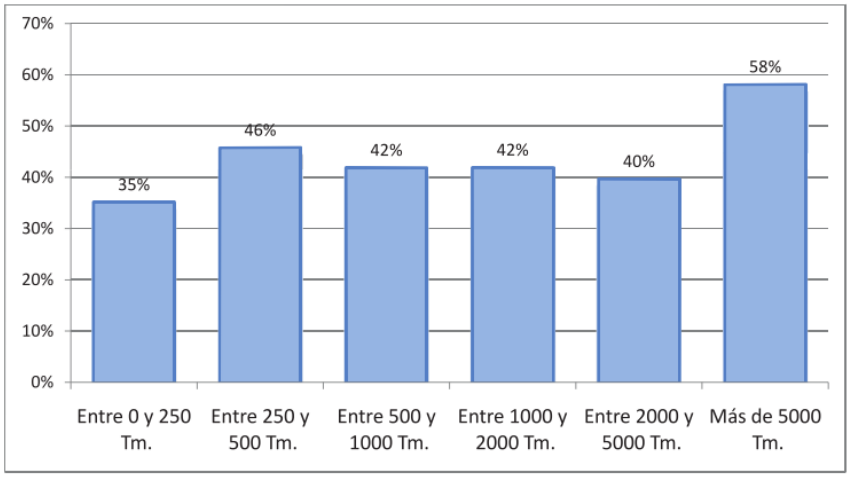

Figura 16. Implementación de Tecnologías en Industrias de España

(Fuente: [31])

Inclusive conociendo que los empresarios encuentran en la automatización y control una herramienta en la mejora del rendimiento industrial y calidad del producto obtenido [44].

En este sentido, hay áreas que necesitan desarrollarse aún más para que la agricultura de precisión pueda consolidarse como una solución amplia y plenamente viable, para todos los segmentos de la agricultura.

Se destacan dos grandes áreas de trabajo:

1. El desarrollo de sensores que permitan obtener, en tiempo real, de forma eficiente y confiable, la deficiencia nutricional o de estrés hídrico de la planta durante su desarrollo para aplicación y corrección en el tiempo preciso y;
2. El desarrollo de dispositivos, programas de computación y estrategias que posibiliten una mayor integración de los datos obtenidos, facilitando así la interpretación y análisis de los mapas y haciendo también más efectivo el manejo localizado. En este sentido, se encuentran trabajos donde relacionan diferentes tipos técnicas y herramientas tecnológicas, entre ellos: Torres Galindo (2015) indica que un sistema de gestión de cultivo basado en AP busca básicamente responder los siguientes cuestionamientos: Localización: ¿Dónde está?; Condición: ¿Cómo se encuentra?; Tendencia: ¿Que ha cambiado desde…; y para dar respuesta con bases sólidas a estos ítems, se tienen algunas herramientas para adquirir y analizar información como pueden ser: los sistemas de teledetección, los sistemas de seguimiento de variables de cultivo – fitomonitoreo, sea con redes de sensores o con sistemas de adquisición de información de campo, los sistemas de recolección y aplicación de insumos en maquinaria agrícola – monitor de rendimiento, banderillero satelital, aplicación de insumos en tasa variable, los sistemas globales de navegación, el software especializado y los sistemas de gestión de cultivo entre los que se encuentran los (SIG) [45].

Para la comunicación de los elementos de un sistema automático se pueden usar diferentes alternativas como es física o inalámbrica, autores como Capraro et al., (2010) implementa un método a través de red RS – 485.

No obstante, en algunos trabajos se relaciona algunos conceptos de los sistemas WSN (Wireless Sensor Networks) como por ejemplo, que es un sistema compuesto Tx/Rx (Transceptores) RF, sensores, microcontroladores y fuentes de poder, que por lo general son auto-organizables, auto-configurables, auto-diagnosticables y auto-reparables [46].

Así mismo, lo definen como redes inalámbricas compuestas por nodos, que, a su vez, están constituidos por sensores y tarjetas de adquisición y una “mote” (tarjeta donde está el procesador y el transmisor). Incluso, resaltan la importancia respecto a la robustez, flexibilidad, eficiencia en el consumo de la energía y el bajo costo, que deben tener los elementos periféricos, así, como las tarjetas de adquisición de datos, y de control.

Igualmente, De León Mata, Pinedo Álvarez, & Martínez Guerrero (2014), menciona que la aplicación de estas técnicas proporciona una base fuerte en el empleo de metodologías que permiten monitorear los patrones o variabilidad espacial y temporal de las actividades antrópicas.

Hoy día, se tiene gran variedad de estándares de comunicación inalámbrica como LAN inalmabrica, IEEE 802.11B (WIFI), PAN inalámbrico, IEEE 802.15.1 (Bluetooth) e IEEE 802.15.4 (ZigBee).

No obstante, algunos autores como Espinosa-Faller & Rendón-Rodríguez (2012) sugieren el estanadar IEEE 802.15.4 (ZigBee), como el más adecuado para aplicaciones de AP.

Por otro lado, Sinha, Wei, & Hwang (2017) mencionan que la conectividad de radio de corto alcance ampliamente instalado como por ejemplo, Bluetooth y ZigBee, no son adecuados para los escenarios que requieren un rendimiento a largo plazo con bajo ancho de banda.

Es así como soluciones M2M *(Machine to machine)* basadas en la tecnología celular pueden proporcionar una cobertura amplia, pero que consumen energía excesiva. Los sistemas IOT *(Internet of Things)* ofrecen una mejor solución para hacer frente a la enorme cantidad de dispositivos en constante evolución de los requisitos subyacentes, tales como cobertura, fiabilidad, latencia, y la rentabilidad.

En este sentido, las redes de baja potencia y área amplia, LPWAN (en inglés, *Low Power Wide Area Network)* se dirigen a estas nuevas aplicaciones y mercados. LPWA *(Low Power Wide Area)* es un término genérico para un grupo de tecnologías que permiten comunicaciones de área amplia en los puntos, de menor costo y reducido consumo de energía.

Es perfectamente adecuado para las aplicaciones de IOT que sólo necesitan transmitir pequeñas cantidades de información en un largo alcance.

Muchas de las tecnologías LPWA representadas en la Figura 13, han surgido en los mercados con y sin licencia, como LTE-M, SigFox, de largo alcance (LoRa), y de banda estrecha (NB) -IoT.

Entre ellos, Lora y NB-IO son las dos principales tecnologías emergentes, que implican muchas diferencias técnicas.

Las diferentes características de los protocolos mencionados se pueden identificar y comparar más fácilmente en la figura 17.

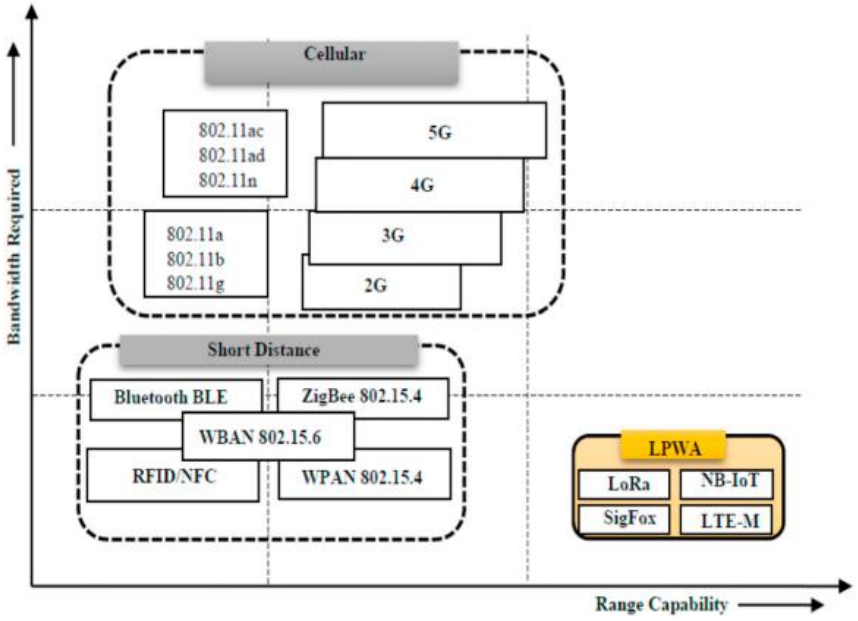

Figura 17. Anchura de banda requerida Vrs Capacidad de rango de distancia corta, celular y LPWA

(Fuente: [38])

2.5. Visión Artificial

2.5.1. Introducción

El desarrollo de sistemas de visión artificial o por computador se ha incrementado últimamente debido a las necesidades que han surgido en la industria y en la agricultura en materia de automatizar y mejorar procesos que normalmente eran desempeñados por humanos como por ejemplo la detección, reconocimiento y clasificación de objetos [50].

En este sentido, la agroindustria ha implementado en los últimos años esta tecnología denominando a estos procesos agricultura de precisión, no obstante, esta detección y reconocimiento de patrones normalmente es realizado bajo ambientes controlados en líneas de producción para medir la calidad de un producto, en este trabajo implementaremos la visión artificial en un ambiente simulado análogo a un ambiente abierto con iluminación irregular, con el fin de identificar enfermedades en los cultivos del lulo.

La visión artificial en este caso servirá para identificar las manchas de Alternaria que se encuentran en las hojas del cultivo mencionado anteriormente.

2.5.2. Definición

La visión artificial o visión por computador es una rama asociada a la inteligencia artificial, que consta de teorías, técnicas y métodos que permite simular de cierta forma el proceso biológico de los humanos, de captar y analizar automáticamente información de imágenes. La visión artificial permite crear algoritmos o códigos para interpretar el significado de una imagen.

La interpretación de ciertos tipos de imágenes, como paisajes, rasgos faciales o señales visuales, es un proceso complejo para la máquina, no obstante, el análisis cuantitativo en las imágenes es relativamente sencillo para los sistemas de visión por computador.

Los procesos de visión artificial se pueden agrupar en tres niveles según su complejidad e implementación, como se pue ver en la figura 18 [50], [51], [52] :

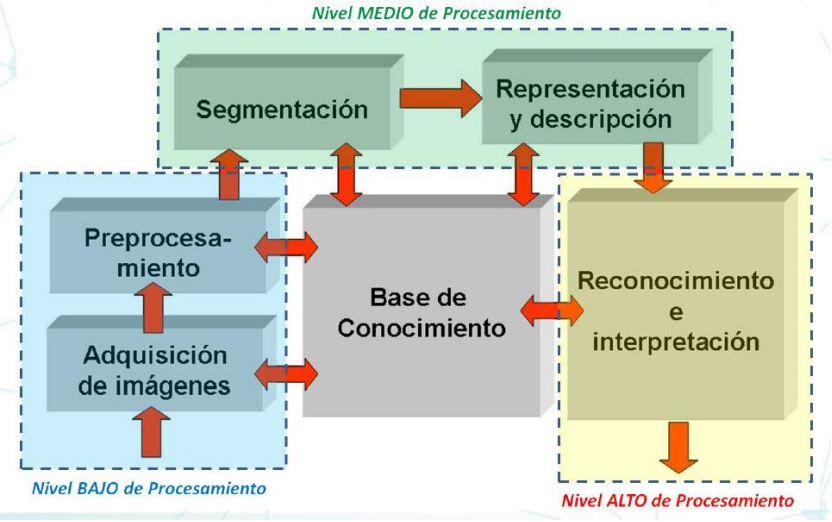

Figura 18. Niveles de Procesamiento de imágenes

(Fuente: [49])

Procesos de nivel bajo: Son operaciones primitivas la captura de imágenes y el preprocesamiento para reducir, ruidos, sombras, brillos, una característica importante de este proceso es que sus entradas y salidas son imágenes.

Procesos de nivel medio: Son operaciones de nivel medio la segmentación y la representación y descripción de objetos individuales presentes en una escena para reducirlos a una forma adecuada para su tratamiento digital, reconocimiento y clasificación de estos. Un rasgo importante de estos procesos es sus entradas son imágenes, pero sus salidas son características extraídas de las imágenes como bordes contornos.

Procesos de nivel superior: son operaciones que reconocen un conjunto de objetos de la imagen y realizan funciones cognitivas que normalmente se relacionan con la visión, conocidos como procesos de reconocimiento e interpretación.

Aunque estas etapas parecen ser secuenciales, no necesariamente lo son, ya que se pueden interrelacionar entre ellas, en la medida que se requiera.

2.5.3. Elementos de los sistemas de visión artificial

2.5.3.1. Sistema de iluminación

La iluminación es de suma importancia en el desarrollo de los sistemas de visión artificial para la obtención de resultados óptimos, entre mejor sea la iluminación más y mejores resultados se podrán obtener a la hora de la obtención y el procesamiento de las imágenes.

Los principales objetivos de la iluminación en la visión artificial son:

- Mantener una constante intensidad

- Mantener fija la dirección del foco de la luz

- Optimizar los contrastes para diferenciar los objetos presentes del fondo.

Los diferentes tipos de iluminación se pueden clasificar según su intensidad, dirección y fuente de origen básicamente.

Según su intensidad: Cuando se varia la intensidad de luz se encuentran diferentes efectos sobre la imagen puede haber más contrastes en algunas zonas como pueden aparecer sombras en distintos lugares en donde se desean o no, por otro lado, una intensidad suave permite apreciar mejor los contrastes, no obstante, se pierden algunos detalles de las texturas.

Según la dirección de la iluminación:

- Posterior: También llamada retroalimentación, consiste en colocar el objeto entre la cámara y la fuente de luz. Este tipo de iluminación es adecuado para el reconocimiento y medida de objetos a través de la detección de sus bordes, esto debido a que se consigue un alto contraste entre los objetos y el fondo, aunque se pierden algunos detalles de la escena.
- Frontal: Consiste en iluminar el objeto de forma frontal, es decir, la fuente de luz se ubica al frente del objeto para que los rayos de luz incidan directamente a él, permite visualizar características externas de los objetos como son forma, color o superficies, que permiten una mejor segmentación y reconocimiento de patrones. Es el más usado en visión

artificial, sin embargo, a veces no se obtienen un buen contraste entre el objeto y el fondo, debido a la aparición de sombras y reflejos.

- Direccional: Consiste en ubicar la fuente de luz en algún sentido para destacar determinadas características del objeto, la dirección del foco está hacia el objeto, este tipo de iluminación permite destacar detalles en algunos sitios puntuales del objeto, se usa principalmente en localización y reconocimiento de objetos, inspección de superficies, seguimiento de cordones de soldadura, etc.
- Estructurada: Consiste en proyectar patrones modulados de iluminación sobre el objeto y adquirir información sobre la superficie del objeto utilizando la luz reflejada. Generalmente, la luz son láseres y los usos de este tipo de iluminación son reconstrucciones 3D de objetos y reconocimiento de formas [50].

Según la fuente de iluminación: Existen diversas opciones para llevar a cabo la iluminación necesaria para realizar visión artificial, a continuación, enunciaremos las fuentes más básicas hasta las más actuales y novedosas.

- Lámpara incandescente: fue la primera fuente de origen de luz por energía eléctrica, no obstante, su alto consumo de energía eléctrica y el calor que emitía permitió que fuera reemplaza por otras fuentes de luz con menor consumo de energía y que no emitían calor.
  – Fluorescente: Proporciona una luz brillante sin sombras, pero por su limitada variedad de formas, es limitada para la aplicación en sistemas de visión artificial. Este tipo de iluminación no consume tanta energía como las lámparas incandescentes, sin embargo, su efecto de parpadeo limita también su uso para el procesamiento de imágenes.
- LED:
- Fibra Óptica
- Láser: [50], [53]

En la figura 19, se muestra los esquemas de estos tipos de iluminación, donde se puede apreciar la ubicación de la fuente de luz.

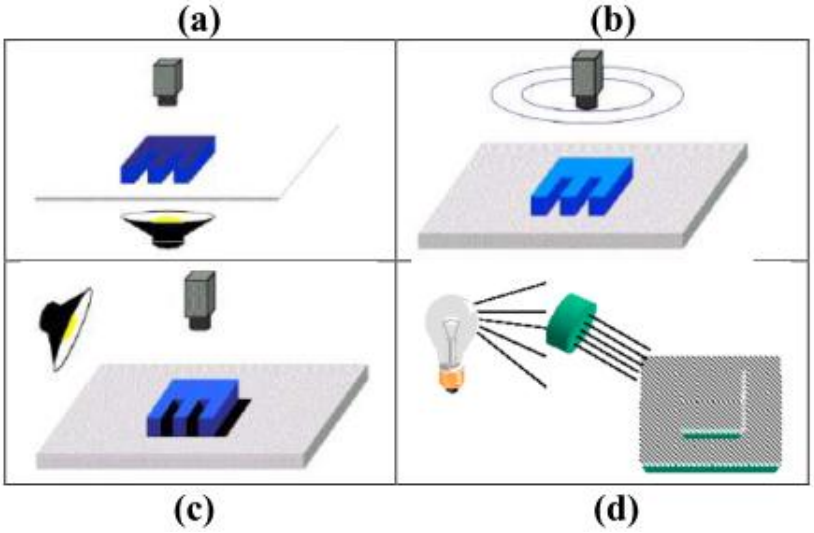

Figura 19. a) Iluminación posterior, b) Frontal, c) Direccional y d) Estructurada

(Fuente: [48])

2.5.3.2. Cámaras y Tarjeta de Captura

Son los elementos que capturan y digitalizan la imagen para después procesarla en el ordenador.

2.5.3.3. Hardware y software

2.5.3.3.1 Hardware

Los hardware son los elementos electrónicos necesarios que permiten desplegar el software, dentro de estos hardware encontramos los ordenadores, tarjetas de procesamiento, controladores y tarjetas de comunicación, entre otros.

2.5.3.3.2 Software

Para realizar proyectos de visión artificial usando Raspberry hay muchos sistemas operativos y ficheros que pueden servir para este fin, debido a que esta tarjeta permite trabajar con sistemas operativos y software open acces, no obstante, es recomendable usar los sistemas operativos (S.O), recomendados en la página oficial del hardware.

En la figura 20, se muestran los softwares o sistemas operativos recomendados desde la plataforma de Raspberry Pi (R-Pi).

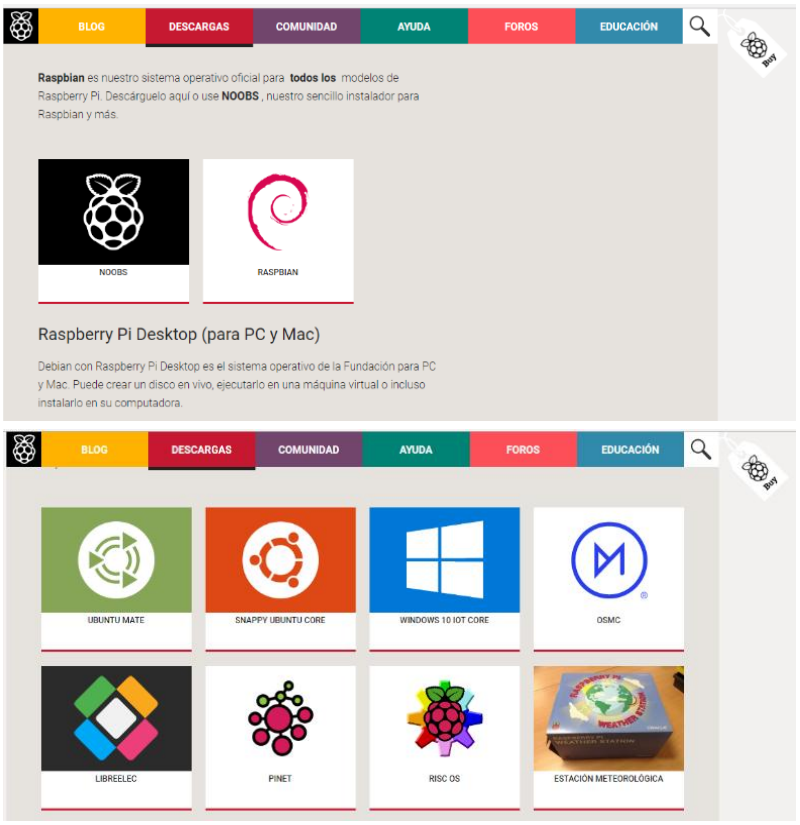

Figura 20. Software y sistemas operativos usados en las RASPBERRY Pi

(Fuente: Propia)

El sistema operativo más usado actualmente para trabajar con este tipo de tarjetas es el Raspbian, debido a que es el sistema operativo oficial de la casa fabricante de la tarjeta, adicionalmente, es también open acces.

Este sistema operativo es basado en Debian, una distribución de Linux optimizado para el hardware de R-Pi, incluye más de 35.000 paquetes que permiten el mayor rendimiento posible, software recompilado y una interfaz sencilla.

Por otro lado, para poder realizar el procesamiento de imágenes y el reconocimiento de patrones usaremos OpenCV, que es una biblioteca de visión por computador de código abierto en ingles *Open Source Computer Vision Library*, licenciada por BSD del Ingles *Berkeley Software Distribution*, la cual, incluye cientos de algoritmos de visión artificial, inicialmente, desarrollada por Intel en 1999 y su última versión se lanzó en junio de 2015, durante los últimos años ha sido actualizado y se le han corregido errores.

OpenCV tiene interfaces para trabajar C/C++, Python y Java, y soporta Windows, Mac OS, GNU/Linux, iOS y Android, la biblioteca está escrita en C/C++ orientado a la eficiencia computacional con un enfoque en aplicaciones en tiempo real.

OpenCV tiene una estructura medular, lo que significa que incluye varias carpetas compartidas los módulos que normalmente se encuentran incluidos en el paquete son:

Core functionality (Funcionalidad básica): En este módulo se definen las definiciones de las estructuras básicas, incluyendo la Matriz multidimensional 'Mat' y funciones básicas usadas por el resto de los módulos.

Image processing (Procesamiento de imagen): En este módulo se incluyen filtros de imagen lineales y no lineales, transformaciones geométricas de imagen, conversiones del espacio de colores, histogramas, etc.

Video (Vídeo): Un módulo para el análisis de vídeo que incluye estimación de movimiento, substracción de fondo y algoritmos de seguimiento de objetos.

Calib3d (Calibración 3D): Algoritmos básicos de geometrías de múltiple vista, calibración simple y estéreo, estimación de la posición de objetos, algoritmos de correspondencia estéreo y elementos de reconstrucción 3D

Features2d (Características 2D): Detectores de características salientes, descriptores y descriptores de coincidencias.

Objdetect (Detección de objetos): Algoritmos de detección de objetos e instancias de clases predefinidas (por ejemplo: caras, ojos, personas, coches, etc.).

Highgui: Una interfaz de uso fácil para simplificar la UI (Interfaz de usuario, del inglés: User Interface).

Videoio: Una interfaz de uso fácil para captura de vídeo y códecs (codificador-decodificador) de vídeo.

Gpu: Algoritmos de diferentes módulos de OpenCV con aceleración de GPU (Unidad de Procesamiento de Gráficos, del inglés: Graphics Processing Unit) [54]

En la figura 21, se presenta la distribución de algunos módulos o librerías que se encuentra dentro del paquete de OpenCV [53].

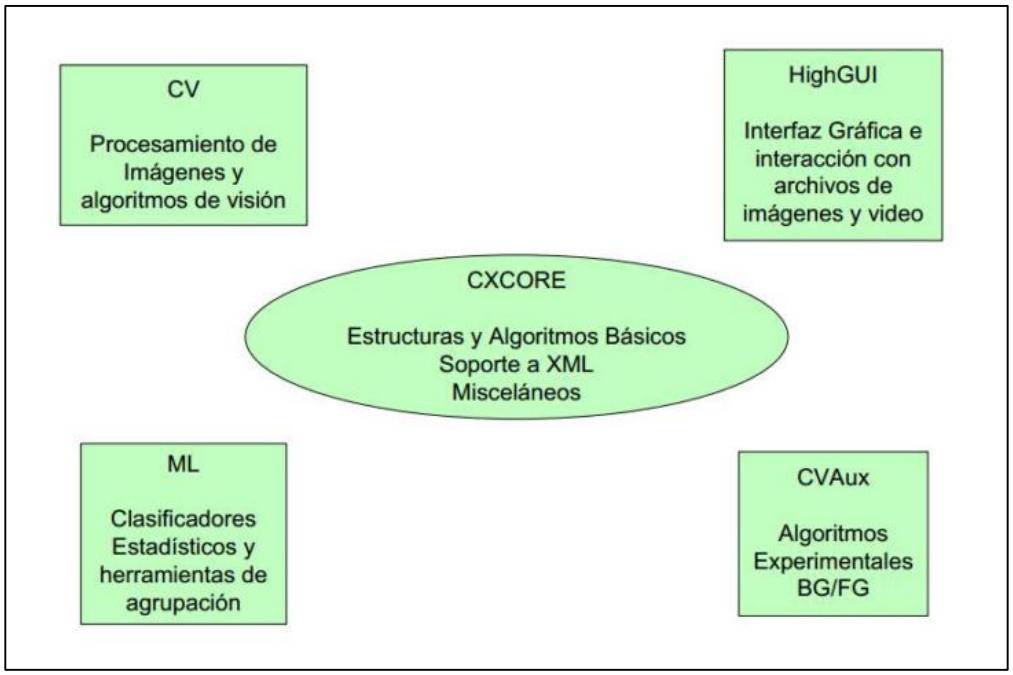

Figura 21. Distribución de librerías dentro del paquete de Open CV

(Fuente: Tomada de la Memoria del trabajo de fin de Máster "Estudio de viabilidad de un sistema basado en Raspberry Pi para aplicaciones de Inspección industrial por Visión Artificial", David Silva Montemayor, pág. 48 [51])

2.5.4. Etapas

Para el procesamiento de imágenes o visión artificial, se deben seguir una serie de etapas, las cuales, cada una cumple una función importante y especifica:

2.5.4.1. Adquisición

La adquisición es el proceso mediante el cual se obtiene información de una escena mediante un equipo óptico, no obstante, dentro de este proceso deben relacionarse dos elementos importantes como es el objeto sensible a la energía irradiada por el objeto (espectro de energía electromagnética) queremos obtener la imagen y el digitalizador que es un elemento electrónico que convierte la salida del dispositivo de detección física en forma digital.

El resultado final de esta imagen es una matriz bidimensional de valores enteros, es importante realizar esta etapa de forma eficaz, para evitar un complique superior en la etapa de procesamiento, por ello se realiza varias pruebas de adquisición

En la figura 22, se presenta un ejemplo del proceso de adquisición de imagen digital [52].

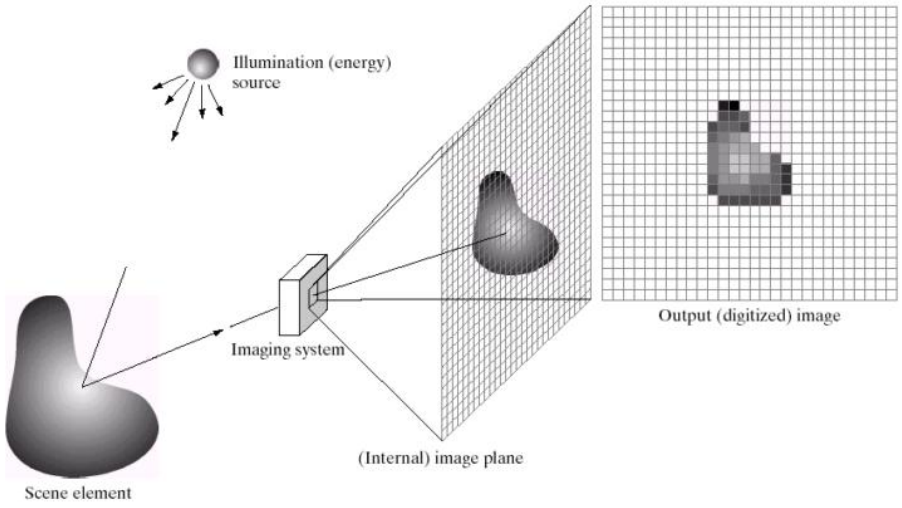

Figura 22. Proceso de Adquisición de imagen digital

(Fuente: [50])

El espectro electromagnético es una representación de energía en función de la frecuencia, la energía viaja a la velocidad de la luz en forma de ondas y se detecta a través de la interacción con el medio ambiente.

En un extremo del espectro están las ondas de radio con longitudes de onda miles de millones de veces más grandes que las de la luz visible. En el otro extremo del espectro se encuentran los rayos gamma con longitudes de onda millones de veces más pequeñas que las de la luz visible. Las ondas electromagnéticas son ondas sinusoidales propagadas de longitud de onda λ, o pueden ser consideradas como una corriente de partículas sin masa que viajan en forma de onda a la velocidad de la luz haciendo referencia al comportamiento dual [52].

El espectro electromagnético se puede expresar en términos de longitud de onda, frecuencia y energía:

La longitud onda λ y la frecuencia ν están relacionadas por la expresión $\lambda = c/\nu$, donde c es la constante de velocidad de la luz (2.998×10^8 m/s).

La energía de los diferentes espectros electromagnéticos está dada por la ecuación $E = h\nu$, donde h es la constante de Planck.

La gama de colores que percibimos en la luz visible es una porción muy pequeña, desde aproximadamente 0.43μm (violeta) hasta aproximadamente 0.79μm, el espectro de color visible

se divide en seis regiones como son: violeta, verde, amarillo, naranja y rojo. Cada sesión se intercala con la siguiente.

Una forma de describir la calidad de una fuente de luz cromática es utilizando tres cantidades básicas: radiancia, luminancia, y el brillo. Radiancia es la cantidad total de energía que fluye de la fuente de luz medida en vatios (W), luminancia es la cantidad de energía que un observador percibe a partir de una fuente de luz medida en lúmenes (lm), y el brillo es un descriptor subjetivo de percepción de la luz [52].

No obstante, en fotometría también se tiene otro concepto o variable de interés como es la iluminancia, definida como el flujo luminoso recibido por una superficie. Su símbolo es E y su unidad es el lux (lx) que es básicamente la relación lm/m^2. Esta variable se puede medir con unos instrumentos llamados luxómetros, que permite tener una idea respecto a la iluminación que se tiene en una superficie.

En la figura, 23 se muestra el espectro electromagnético (energía por fotón) que va desde los rayos gamma (más alta energía) hasta ondas de radio (energía más baja).

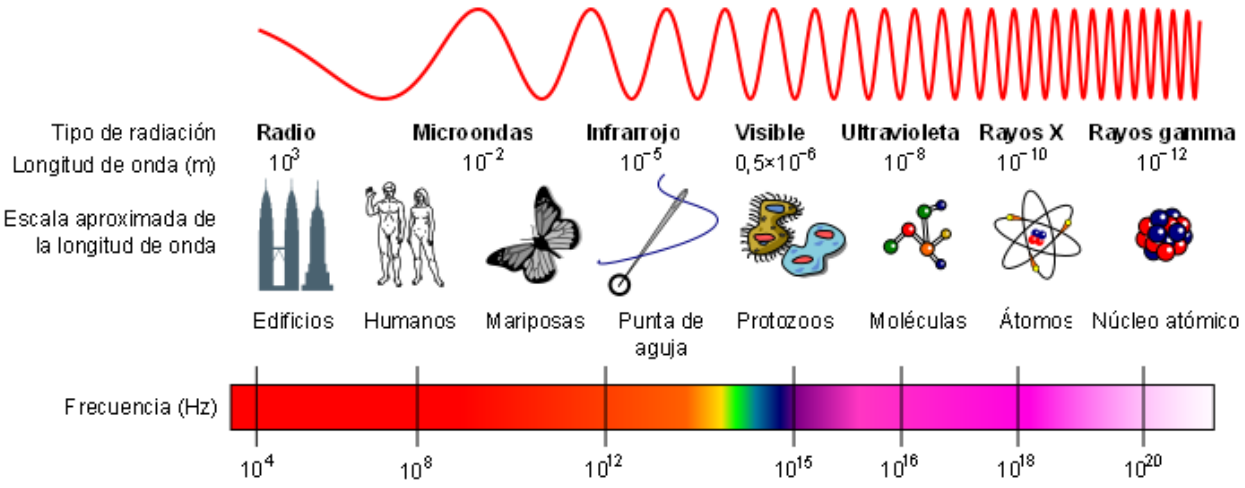

Figura 23. Espectro electromagnético

(Fuente: http://www.quimicafisica.com)

Las imágenes son generadas por la combinación de una fuente de “iluminación” y el reflejo de una absorción de energía provocada por los elementos de la escena que se va a examinar. La energía entrante se convierte en una tensión por la combinación de la energía eléctrica de entrada y el material del sensor, que es sensible al tipo particular de energía que se detecta [52].

Los sensores son dispositivos que modifican una señal eléctrica en función de la intensidad luminosa que perciben, generalmente, la salida de estos sensores es una onda de tensión continua cuya amplitud corresponde al fenómeno físico que está detectando. Para crear una imagen digital, se digitaliza los datos u ondas continúas detectadas por los sensores. Se da entonces la descomposición de la imagen real en una matriz discreta de puntos de un determinado tamaño.

Para entender mejor el proceso de digitalización, se debe comprender primero dos conceptos clave, el muestreo y la cuantificación. El proceso de digitalización de una imagen se debe realizar en sus coordenadas (ejes x, y) y amplitud. El muestreo es la digitalización de los valores en las coordenadas y la cuantificación es la digitalización de los valores de la amplitud. La cantidad del número de muestras y de niveles amplitud o intensidad, determinan la calidad de la imagen.

En la figura 24, se ilustra el proceso de digitalización de una imagen.

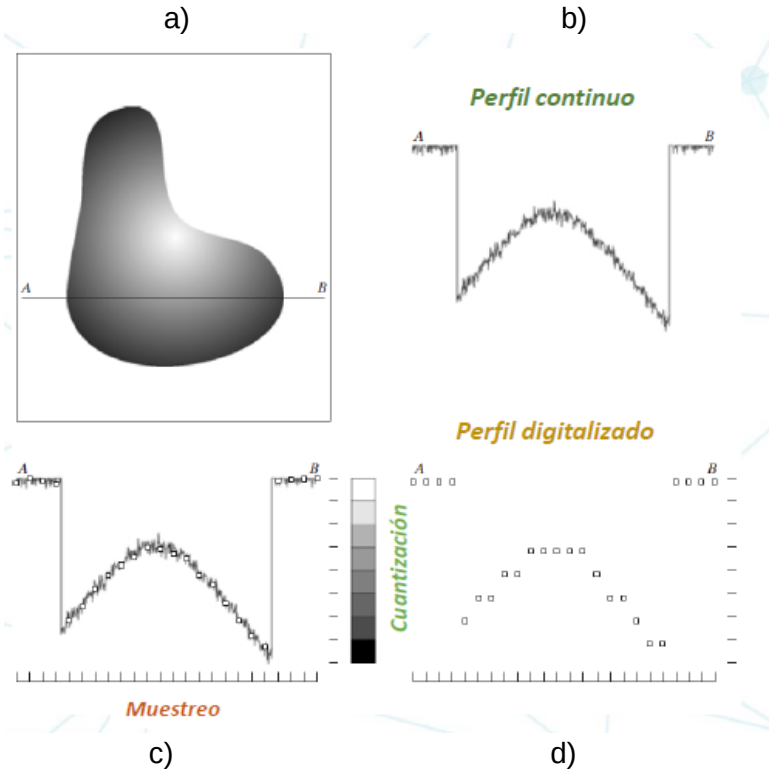

Figura 24. Generación de una imagen digital.

(a) Imagen continúa. (b) Línea de escaneo de A a B en la imagen continua (nivel de intensidad), utilizada para ilustrar los conceptos de muestreo y cuantificación. (c) Muestreo y cuantificación. (d) Línea de escaneo digital

(Fuente: [50])

Igualmente, en la figura 25, se muestra el proceso de muestreo y cuantificación de una imagen.

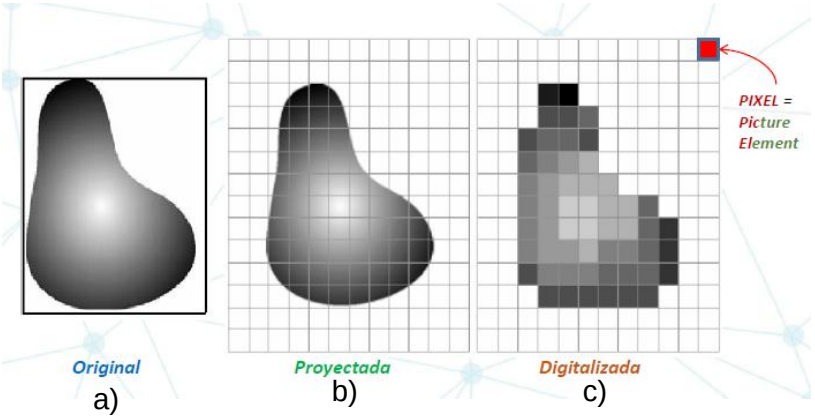

Figura 25. (a) Imagen original (b) Imagen continúa proyectada sobre un arreglo de sensores. (c) El resultado del muestreo y cuantificación de imagen

(Fuente: [50])

Como se dijo anteriormente, para digitalizar una imagen se realiza a través de una matriz con coordenadas X y Y.

Sea entonces f(s, t) que representa la imagen bidimensional (2D) como una función de dos variables continuas, s y t. El valor o amplitud de f en las coordenadas espaciales (x, y) es una magnitud escalar positiva cuyo significado físico es la intensidad o nivel de gris de la imagen. Convertimos esta función en una imagen digital mediante el muestreo y la cuantificación, resultando la función f(x, y), que contiene M filas y N columnas, donde x e y son las coordenadas discretas. Por razones de claridad y conveniencia de notación, utilizamos valores enteros para estas coordenadas discretas: y = 0, 1, 2,…, N −1 e y = 0,1,2,…, M −1. Se considera la imagen como digital cuando las coordenadas y los valores de intensidad son cantidades discretas finitas [52].

En forma de ecuación se representa a través de una matriz numérica de M x N, como:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots f(0,N-1) \\ f(1,0) & f(1,1) & \dots f(1,N-1) \\ \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots f(M-1,N-1) \end{bmatrix}$$

Es decir, se representa por una matriz de números reales y cada elemento de esta matriz se denomina elemento de imagen o pixel (picture element).

Debido al almacenamiento digital de los diferentes hardware los niveles de intensidad y cuantificación se manejan como una potencia entera de 2 ($L=2^k$, en donde k es el número de bits).

En imágenes de escalas de grises la intensidad se representa habitualmente por 8 bits (1 byte), con esta dimensión se puede codificar 2^8=256 tonos de gris (0 a 255), la correspondencia entre el valor digital y el tono es el siguiente: para el negro 0 y para el blanco 255.

Para realizar un cálculo rápido respecto al tamaño que puede tener una imagen de acuerdo a la cantidad de pixeles (cuantificación) y a la cantidad de tonos de grises (intensidad), se aplica la siguiente ecuación b=M x N x k, en este sentido si tenemos una imagen de 640 x 320 pixeles con 256 tonos o niveles de grises, el tamaño digital de esta imagen seria 640 x 320 x 8 = 1.638.400 bits de memoria = 204.800 bytes, en las imágenes de colores es similar solo que se necesita tres veces más de espacio para su almacenamiento debido a las tres matrices que se generan (R G B).

Un concepto importante que tratar en el proceso de la adquisición de imágenes y su posterior digitalización es la vecindad de los pixeles, esta vecindad es la relación que existe entre un pixel y los demás pixeles cercanos a él, los vecinos del pixel se derivan del enmallado que se haya usado al momento de la digitalización

En un enmallado cuadrangular un pixel p con coordenadas x e y, tiene cuatro pixeles vecinos directamente con coordenadas (x+1, y), (x-1, y), (x, y+1), (x, y-1), este conjunto de pixeles se llaman 4 – vecinos de p, y se denotan como $N_4(p)$, igualmente, las coordenadas de los cuatro vecinos diagonales son (x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1), y se notan como $N_D(p)$, estos pixeles junto con los 4 – vecinos de p, se llaman 8 – vecinos de p y se denotan como $N_8(p)$ [52].

En la figura 26, se presentan los pixeles 4 – vecinos de color rosa con sus respectivas coordenadas y de color azul se presentan los pixeles diagonales también con sus respectivas coordenadas.

P(x-1, y+1)	P(x, y+1)	P(x+1, y+1)
P(x-1, y)	P(x, y)	P(x+1, y)
P(x-1, y-1)	P(x, y-1)	P(x+1, y-1)

Figura 26. Representación de los pixeles 8 - vecinos

(Fuente: Propia)

2.5.4.2. Procesamiento

El procesamiento o procesado de una imagen es un conjunto de técnicas o transformaciones que permite de cierta forma obtener información relevante de algunas características de la escena o imagen o simplemente para mejorar su calidad.

El procesamiento de imágenes va desde simple operaciones aritméticas o geométricas entre pixeles, hasta operaciones de filtrado con transformadas como Fourier, Canny o Sobel. A continuación, relacionan algunas operaciones que utilizan para procesar imágenes:

1. Operaciones aritmético-lógicas
2. Operaciones Geométricas
3. Operaciones sobre el histograma
4. Filtrado en el dominio de la frecuencia
5. Filtrado en el dominio del espacio
6. Operaciones Morfológicas

Operaciones aritmético-lógicas

Las operaciones aritmético – lógicas son las técnicas de procesamiento más comunes que se le hacen a las imágenes y radica en la operación de pixel a pixel de la imagen, dentro de este grupo de operaciones podemos encontrar las siguientes [5]:

Conjunción, disyunción, negación, suma, resta, multiplicación, división, entre otras.

En la figura 27, se muestren algunos ejemplos de la aplicación de estas operaciones.

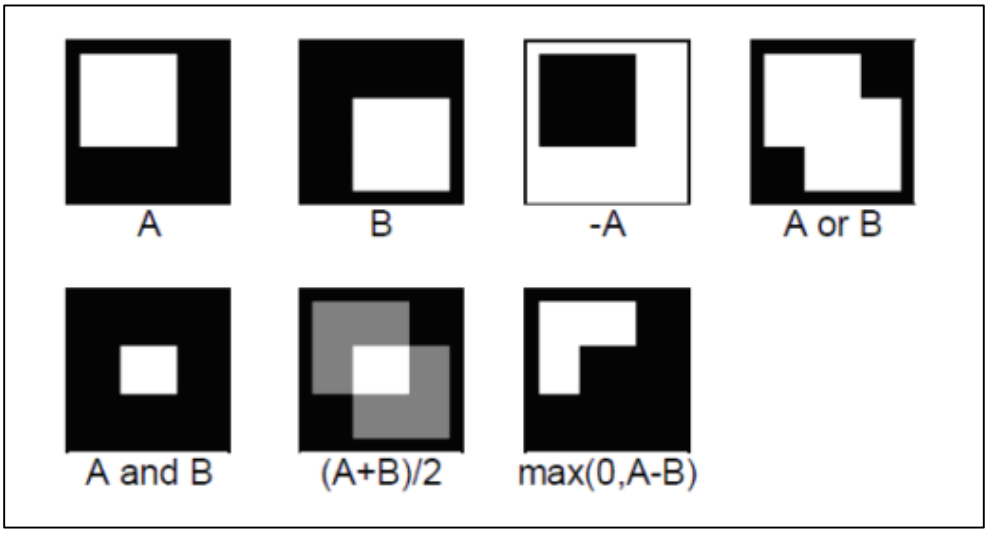

Figura 27. Operaciones aritmético – lógicas

(Fuente: Tomada del trabajo de fin de grado “Visión por computador para UAS”, Mónico Gonzales, pág. 11 [5]

Operaciones Geométricas

Son operaciones en las que se modifica la ubicación o posición de la imagen, estas se pueden expresar como una multiplicación de matrices las más frecuentes son:

Traslación, rotación, escalado.

En la figura 28, se ilustran algunas de es estas operaciones geométricas.

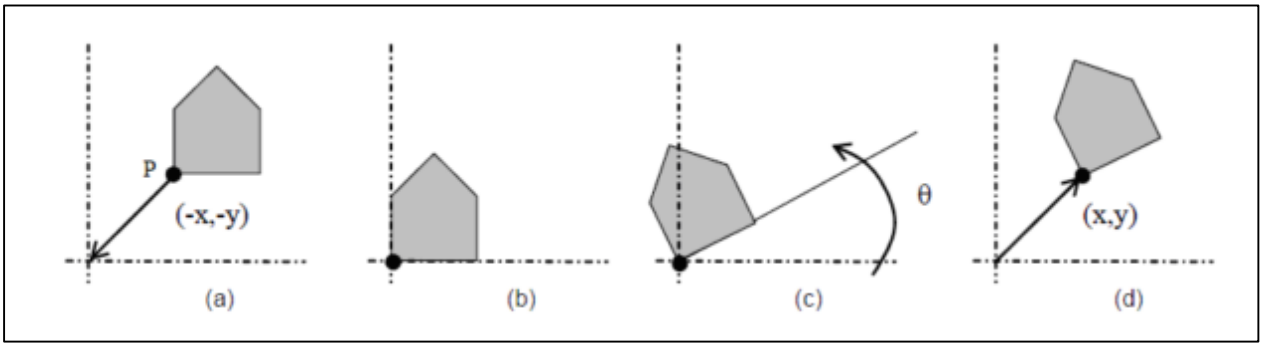

Figura 28. Operaciones Geométricas de una imagen

(Fuente: Tomada del trabajo de fin de grado “Visión por computador para UAS”, Mónico Gonzales, pág. 11 [5])

La operación (a) es una traslación al origen. La operación (b) define el punto de rotación de la imagen. La operación (c) rota la imagen respecto el origen de coordenadas y la operación (d) vuelve a trasladar la imagen, ya girada, al punto inicial [5].

Operaciones sobre el histograma

Las operaciones sobre el histograma de la imagen, permite modificar el contraste y la luminosidad de la imagen, para entender mejor este concepto, explicaremos en que consiste el histograma.

El histograma es un diagrama de barras, en donde en el eje horizontal se encuentran los niveles de cuantización y en el eje vertical corresponde a la cantidad de pixeles que hay en cada nivel de cuantización. Los histogramas se suelen dar para imágenes en blanco y negro con 256 niveles de intensidad (0 – 255), para imágenes en color se le asigna un histograma para cada color de las componentes RGB, el histograma se suele normalizar en una escala entre 0 y 1, para evitar la dependencia con el número de pixeles y niveles.

Las operaciones más comunes en el histograma son:

Ajuste de contraste: El contraste se define como la diferencia de intensidad pronunciada.

En la figura 29, se presentan un ejemplo de una imagen a la cual se le aplico la operación de contraste.

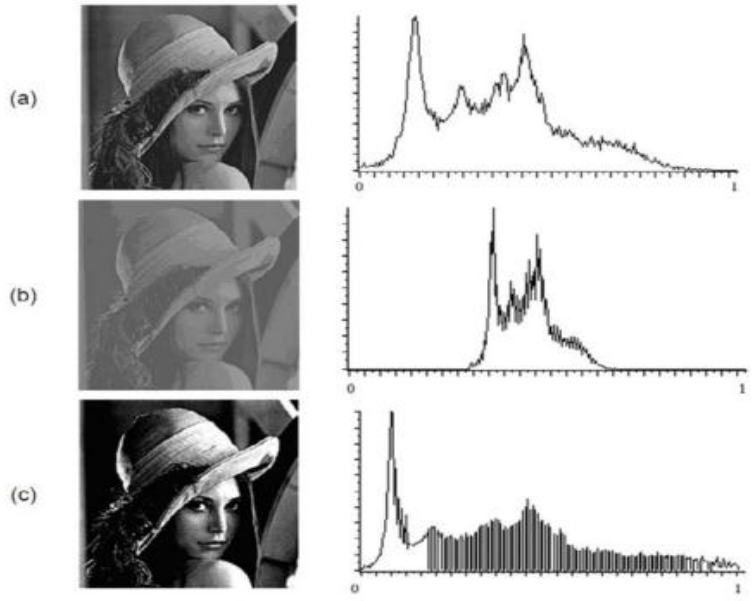

Figura 29. Imagen original (a) frente a imágenes con ajuste de contraste (b) y (c)

(Fuente: [50]

Ecualización del histograma: Consiste en rehacer un nuevo histograma realizando una distribución uniforme de los niveles de intensidad de la imagen.

En la figura 30 y 31, se muestran dos ejemplos de dos imágenes a las cuales se les aplico el proceso de ecualización [55], [56].

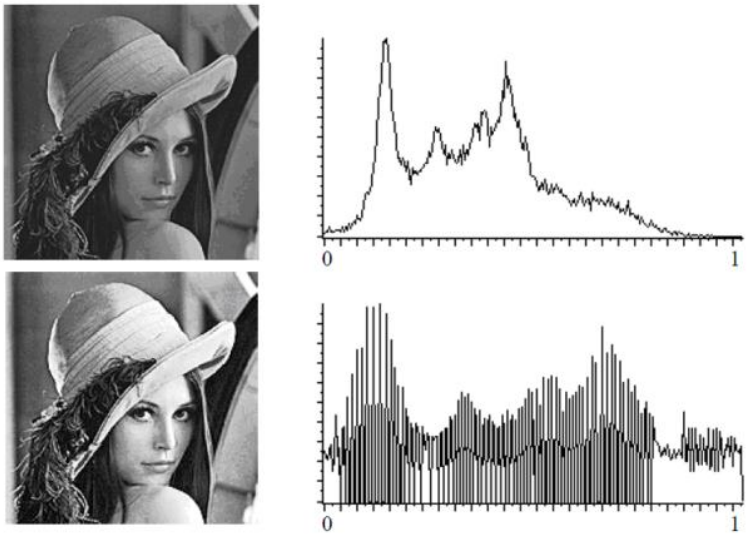

Figura 30. Imagen original (a) frente a imagen con el histograma ecualizado (b)

(Fuente: [50]

Figura 31. Imagen en escalas de grises Vrs imagen ecualizada

(Fuente: [54])

Filtrado en el dominio de la frecuencia:

El filtrado de una imagen en el dominio de la frecuencia procesa una imagen aplicando la transformada de Fourier de la imagen y trabajando sobre el dominio de esta, para ello se aplica el teorema de convolución, es decir, lo primero que se realiza es la transformada de Fourier de la imagen, después se le aplica le filtro y posteriormente se realiza la transforma inversa de Fourier para devolverla al dominio espacial. Las etapas de este proceso se pueden visualizar en la figura 32.

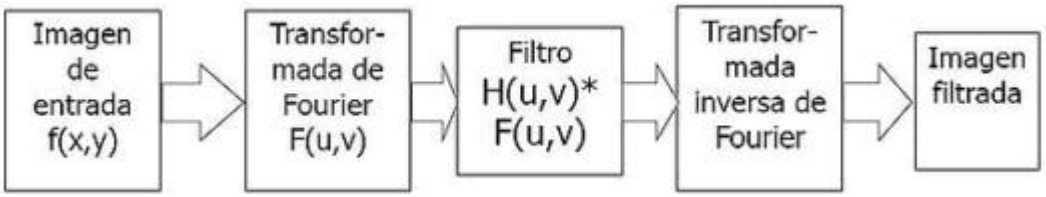

Figura 32. Etapas del proceso de convolución para el filtrado en el dominio de la frecuencia

(Fuente: [50])

No obstante, este no es el único método que se puede aplicar para realizar el filtrado en el dominio de la frecuencia de una imagen, también se puede usar la transformada rápida de Fourier (TRF), la bidimensional o la discreta.

Los filtros más comunes son:

Filtro pasa bajos: Este filtro atenúa las frecuencias altas y deja pasar sin modificación las frecuencias bajas, el resultado es un suavizado en las transiciones de la imagen, reduciendo ruidos, pues atenúan los cambios fuertes de intensidad.

Filtro pasa altos: Este filtro es el caso contrario del anterior, atenúa las frecuencias bajas y deja pasar sin modificación las frecuencias altas, el resultado es una mejor detección de los bordes y un refuerzo de los contrastes de la imagen.

Filtro pasa banda: Este filtro atenúa las frecuencias muy altas y muy bajas, dejando pasar pasar las frecuencias medias.

En la figura 33, se ilustran los respectivos filtros enunciados anteriormente.

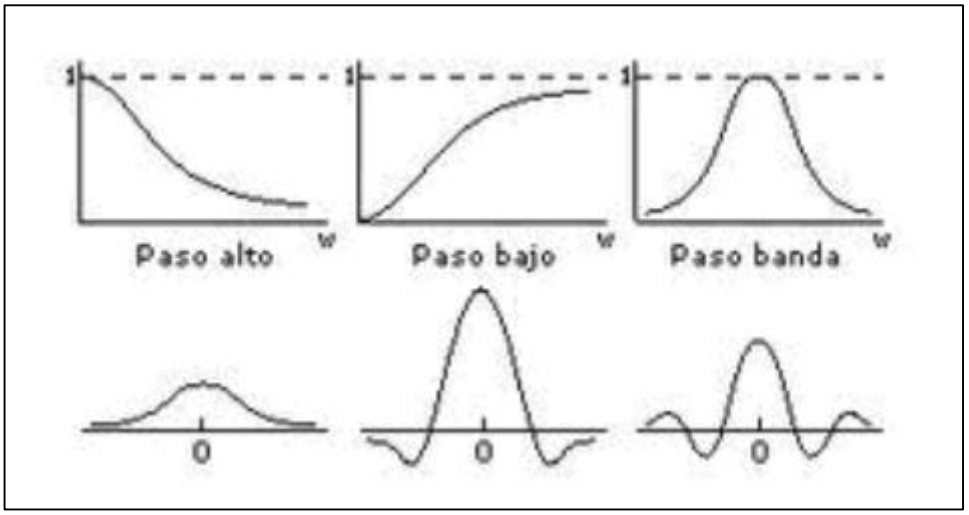

Figura 33. Tipos de filtros en el dominio de la frecuencia.

(Fuente: [50])

Estos filtrados tienen una serie de ventajas, como que son sencillos de implementar, rápidos debido al uso del teorema de convolución, fácil asociación entre tonalidad y frecuencia, igualmente, tiene desventajas como la necesidad de conocer varios campos para el desarrollo de una aplicación de procesamiento de imágenes y la imposibilidad de eliminar totalmente el ruido.

Filtrado en el dominio del Espacio:

El filtrado en el dominio del espacio se realiza directamente sobre la imagen, normalmente, se usa un filtro de convolución, aplicando una máscara de convolución que reduce la operación a multiplicación de los pixeles de la imagen por una matriz de convolución, que será responsable del filtro. Igualmente, hay otros filtros que son los no lineales, que no se basan en la máscara de convolución en cambio realiza una convolución de la máscara sobre la imagen. Se sigue el siguiente Teorema de convolución en el espacio $g(x,y) = h(x,y) * f(x,y)$.

Los filtros espaciales se dividen en los siguientes grupos:

1. Suavisado o Filtro pasa bajos: Se utilizar para eliminar ruidos o detalles de poco interés, dependen mucho de la máscara usada. Pueden ser de promedio, promediando los pixeles contiguos con una matriz unidad; mediana, de media, de pasa bajo frecuencia, el del bicho raro y el gausiano. El filtro de la mediana y el del bicho raro son filtros no lineales, el resto son lineales [5], [55].

En las figuras 34 y 35 se presentan dos ejemplos del filtro de suavizados uno a la foto en escala a grises y el otro aplicado a la foto original.

Figura 34. Imagen en escala de grises Vrs imagen suavizada

(Fuente: Tomada del trabajo de fin de grado “sistema de visión por computador para detectar hierba no deseada en prototipo de cultivo de frijol usando ambiente controlado”, Diego Betancourt, pág. 24 [54])

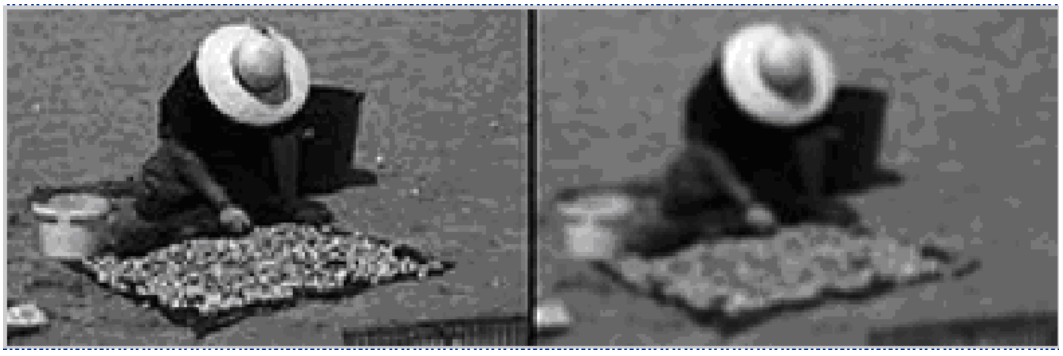

Figura 35. Imagen original Vrs imagen con filtro de suavizado

(Fuente: Tomada del trabajo de fin de grado “Visión por computador para UAS”, Mónico Gonzales, pág. 15 [5])

2. Atenuación o filtro pasa altos: Intensifica los bordes, los detalles y los cambios de alta frecuencia, y atenúa las zonas de tonalidad uniforme. Permite mejor la identificación de objetos en la imagen, y puede propiciar la generación de ruido.

3. Realce de bordes por desplazamiento y diferencia: Sustrae de la imagen original una copia desplazada de la misma. Su resultado realce los bordes existentes según la máscara de convolución empleada sea vertical, horizontal o diagonal (horizontal/vertical).
4. Laplaciana: Este tipo de filtro realiza los contornos en todas las direcciones, ya que trabaja con el operador Laplaciano, permite unos buenos resultados, pero aumenta la generación de ruidos.
5. Filtrado de realce de borde por gradiente direccional: Se emplea para destacar y resaltar con mayor precisión los bordes en una dirección determinada. Trabaja con el cambio de intensidad entre los pixeles contiguos
6. Obtención de contornos: Este filtro de contorno se basa en la diferencia de intensidad que se da de pixel a pixel contiguo. Requiere un menor conste computacional, ya que trabajan desde el operador gradiente de una máscara que hace que se pueda obtener el contorno de la imagen y se puede clasificar en formas existentes de ella, los filtros más utilizados son: el filtro de Sobel, el filtro de Prewitt, el de Roberts y el detector de contornos Canny.

En la figura 36, se presenta un ejemplo de la aplicación de los filtros Canny y filtro Sobel.

Figura 36. Imagen original (Izquierda) frente a imagen tratada con el filtro de Canny (Centro) y tratada con el filtro de Sobel (derecha)

(Fuente: Tomada del trabajo de fin de grado “Visión por computador para UAS”, Mónico Gonzales, pág. 34 [54])

Operaciones Morfológicas:

Las operaciones morfológicas se deben a la morfología matemática, basada en la teoría de conjuntos. En la morfología matemática las imágenes binarias son conjuntos con puntos en dos dimensiones que representan los puntos activos de la imagen. Las imágenes en escala agrises son conjuntos en tres dimensiones, donde la tercera componente es la intensidad del gris.

Puestos que son operaciones basadas en conjuntos sus definiciones también tendrán esta notación. Igualmente, se recomienda trabajar con definiciones de imágenes binarias ya que son más sencillas de trabajar.

A continuación, se enlistarán los diferentes tipos de operaciones morfológicas que se pueden realizar:

1. Dilatación: La dilatación de una imagen A con un elemento estructurante B consiste en examinar cada pixel de A, y si está en valor 0, pasarlo al valor 1 si alguno de sus vecinos está en la imagen A al ser recorrida por la imagen B (o lo que es lo mismo tiene valor 1). Normalmente se utilizan como vecinos los 8 pixeles que rodean al estudiado.
2. Erosión: Es el proceso inverso a la dilatación pasar a valor 0 los pixeles con valor 1 si alguno de sus vecinos está con valor 0.
3. Apertura: Es la operación de erosionar y luego dilatar una imagen.
4. Cierre: Es la operación de dilatar y luego erosionar, por ende, es la operación inversa a la apertura
5. Gradiente Morfológico: Es la operación resultante de restar la imagen erosionada de la imagen dilatada (Gradiente=Dilatación -Erosión).
6. Top Hat: Esta operación consiste en restar a la imagen original la imagen de apertura (Top Hat=Original-Apertura)
7. Black Hat: Esta operación consiste en realizar la resta de la imagen Original a la imagen resultante de Cierre (Black Hat=Cierre-Original), esta operación aísla las zonas oscuras.

En la figura 37, se presenta un ejemplo de la aplicación de las diferentes técnicas de operaciones morfológicas vistas anteriormente.

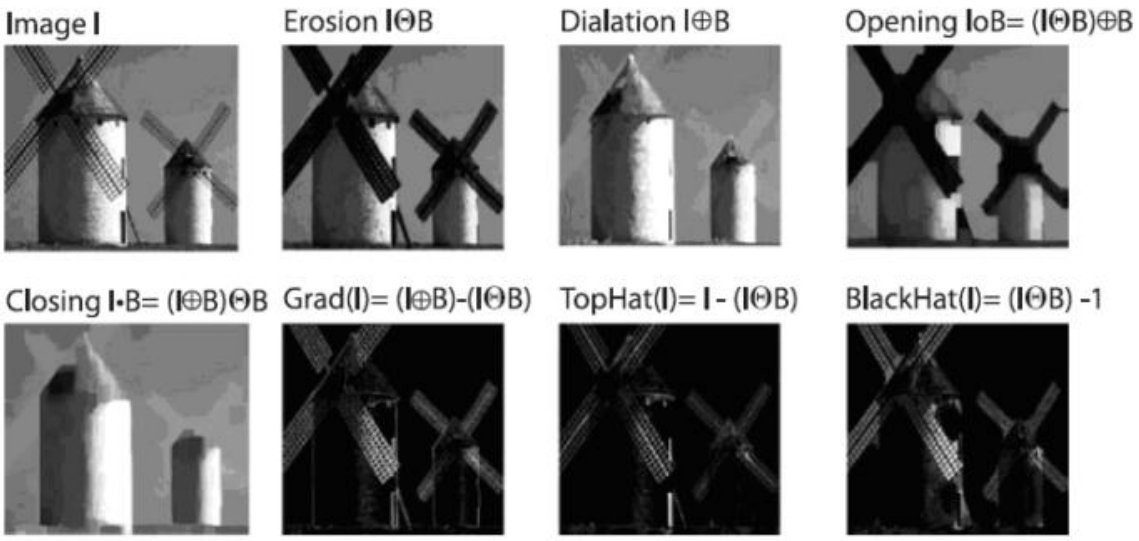

Figura 37. Operaciones Morfológicas que se pueden realizar a una imagen

(Fuente: Tomada del trabajo de fin de grado "Visión por computador para UAS", Mónico Gonzales, pág. 35 [54]

2.5.4.3. Segmentación

La segmentación es la siguiente fase de la visión por computador o visión artificial, consiste en dividir la imagen en regiones homogéneas respecto a una o más características como puede ser el brillo o el color, para facilitar su posterior reconocimiento. Esto se realiza tanto para localizar objetos como para encontrar los bordes de estos en una imagen.

La segmentación asigna una etiqueta a cada pixel de la imagen de modo que los pixeles que comparten esta etiqueta tendrán ciertas características similares visuales. Cuando este proceso logra identificar todos los pixeles con esta etiqueta y los resalta de forma inequívoca con respeto a los otros, se dice que se tiene una segmentación completa, en caso contrario, se contará con una segmentación parcial, y por ende, se deberá hacer un proceso adicional de procesamiento.

En la figura 38, se ilustra un ejemplo de la asignación de etiquetas para la segmentación de imágenes.

Figura 38. Etiquetado de los objetos de una imagen

(Fuente: Tomada del trabajo de fin de grado “sistema de visión por computador para detectar hierba no deseada en prototipo de cultivo de frijol usando ambiente controlado”, Diego Betancourt, pág. 28 [54]

Los algoritmos de segmentación están basados en las discontinuidades de los niveles de gris, segmentando la imagen desde los cambios grandes de niveles de gris entre pixeles, como se hace en los casos de detección de líneas, detección de bordes, detección de esquinas, de puntos aislados.

Existen diversos métodos de segmentación, a continuación, se expondrán los más relevantes y usados a nivel mundial en visión artificial.

1. Método del valor umbral: El método del valor de umbral o umbralización es un método por el que se convierte una imagen de niveles de gris o de color a una imagen binaria, de manera que los niveles de interés tienen una etiqueta diferente al fondo de la imagen. Requiere un coste computacional muy bajo, y por tanto, es una técnica rápida, para ser utilizada en tiempo real en un computador o sistema embebido.

Figura 39. Imagen en escala de gris Vrs imagen binarizada

(Fuente: Tomada del trabajo de fin de grado “sistema de visión por computador para detectar hierba no deseada en prototipo de cultivo de frijol usando ambiente controlado”, Diego Betancourt, pág. 25 [54]

El caso más simple de umbralización es el umbral único o fijo, en el que se establece un umbral fijo sobre el histograma, que marca la separación y es el punto clave del método. El umbral es el límite por el que se divide la imagen, asignando cada pixel a uno de los dos segmentos comparando el nivel con el umbral.

Los métodos de valor de umbral son métodos de segmentación completos, ya que cada pixel pertenece a uno de los dos segmentos y solo a uno de estos segmentos. La imagen final se calcula respecto al umbral t de la siguiente manera:

$$T_{global}(g) = 0 \; si \; g < t$$

$$T_{global}(g) = 1 \; si \; g \geq t$$

Existen variante de este método, como es la umbralización de banda, en el que se coloca un límite inferior y un límite superior; la multiumbralización, que consiste en varios umbrales lo que repercute en varias regiones y no aplica a una imagen binaria, la semiumbralización, consiste en mantener la imagen tal cual a partir del umbral, pero binalizarla si el valor del pixel es inferior al umbral, o la umbralización adaptiva, en la que se permite el cambio del umbral según las características del entorno del punto de estudio.

En la figura 40, vemos el proceso de umbralización de Otsu, el histograma de niveles de gris de una imagen (b), el nivel para realizar la umbralización (c) y la imagen binarizada con este umbral (d).

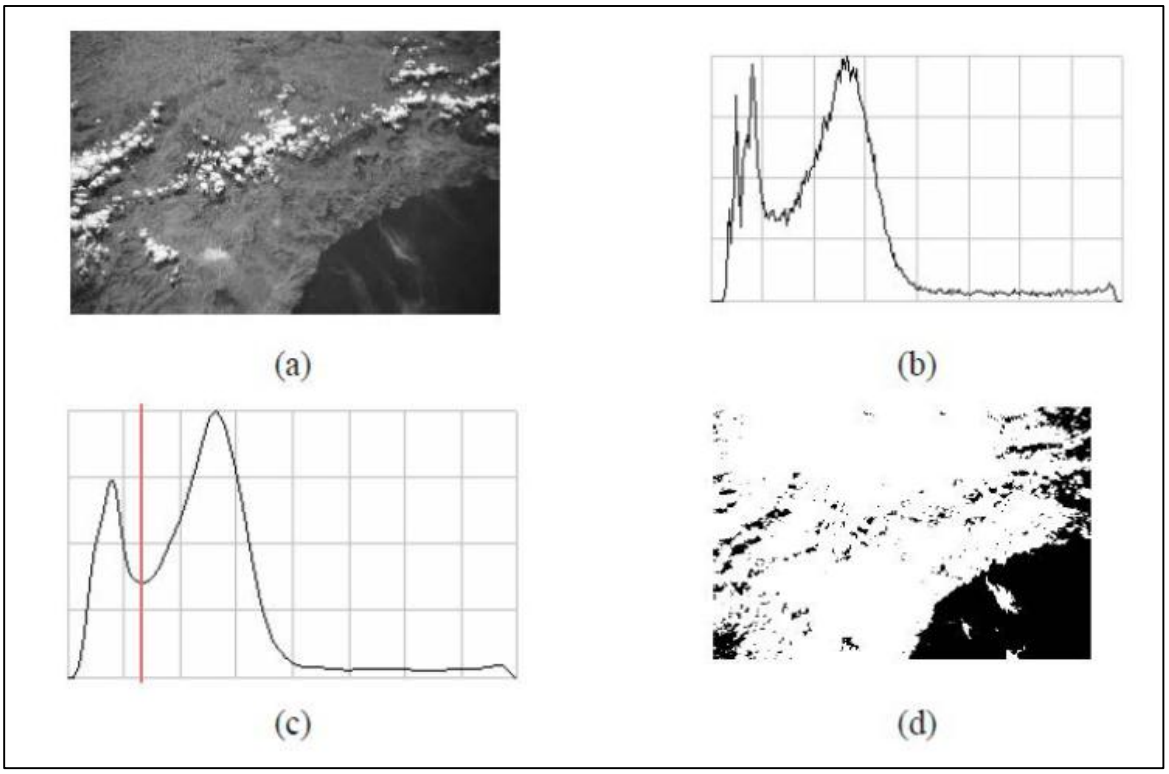

Figura 40. Umbralización de Otsu

(Fuente: Tomada del trabajo de fin de grado “Visión por computador para UAS”, Mónico Gonzales, pág. 37 [54])

La umbralización tiene su explicación en el método matemático de Otsu, ll amado así por su creador Nobuyuki Otsu. Este método utiliza la varianza, calculando el valor de umbral de manera que esa varianza sea la menor posible en cada segmento, pero que sea máxima entre segmentos distintos. Se calcula el cociente de ambas varianzas y se busca un valor de umbral que sea el que maximice ese cociente [56].

2. Transformación de Hough: Es un algoritmo creado en 1962, que permite encontrar ciertas formas como rectas, círculos, etc, dentro de una imagen. La versión más sencilla es la que encuentra las líneas como características en una imagen. Este algoritmo aprovecha la ecuación de la recta que pasa por un punto genérico (x_i, y_i) cuya ecuación es: $(y_i = mx_i + b)$, siendo m y b parámetros variables. Principalmente es un método estadístico y

consiste en que para cada punto que se desea averiguar si es parte de una recta se aplica una operación en cierto rango, con lo que se averiguan las posibles rectas de las que puede ser parte el punto. Esto se realiza para todos los puntos de la imagen y las líneas detectadas son las que tengan mayor cantidad de puntos repetidos, ya que la segmentación se realiza buscando entre los vecinos del pixel analizado.

3. Métodos basados en el histograma: Estos métodos son muy eficientes, ya que solo se requiere una única pasado por los pixeles de la imagen. Una vez se calcula el histograma, los picos y valles de este se usan para localizar los grupos de pixeles en la imagen
4. Clustering o método de agrupamiento: Es una técnica de segmentación iterativa con la cual se divide la imagen en un número determinado de cluster o regiones.

Entre algunos algoritmos que usan estos principios podemos encontrar: El algoritmo de las k-medias que fue inventado en 1956 y el algoritmo de Lloyd.

5. Método de crecimiento de regiones: Es un método que determina zonas dentro de una imagen basándose en criterios de similaridad y proximidad entre los pixeles de esta. En estas técnicas, el criterio de homogeneidad (o no homogeneidad) entre regiones adyacentes es indicador clave para unir o separar regiones de la imagen. Dicha homogeneidad se puede basar en criterios como: el nivel de gris medio, el color, la forma, etc. El resultado de la segmentación es una división de la imagen en zonas homogéneas.
   Las técnicas de segmentación basadas en contornos tratan de encontrar fronteras entre regiones, las técnicas de segmentación basadas en regiones trabajan mejor en imágenes con ruido, en donde los contornos son difíciles de encontrar.
   La segmentación resultante de una detección de contornos y la basada en crecimiento de regiones, aplicadas a una misma imagen no producen siempre el mismo resultado. Es recomendable para mejores resultados combinar los dos tipos de técnicas de segmentación.
6. Método del conjunto de nivel: La propagación de curvas de nivel, es una técnica popular en los análisis de imágenes de extracción de objetos, la reconstrucción de 3D, etc.
   La idea central de esta técnica consiste en desarrollar una curva hacia el menor potencial de una función de coste que en su definición refleja la tarea a la que está dirigida e impone ciertas limitaciones de suavidad. Las técnicas de Lagrange se basan en la parametrización de alguna estrategia de muestreo y luego desarrolla cada elemento de acuerdo con la imagen y a sus condiciones internas. Entre sus limitaciones tenemos: cómo decidir la

estrategia de muestro, la estimación de las propiedades geométricas internas de la curva, el cambio de su topología, etc.

Este método fue propuesto inicialmente para realizar seguimiento de interfaces móviles por Osher y Sethian en 1988 y se ha diseminado a través de vario tipos de imágenes a finales de los noventa.

7. Métodos de particionamiento gráfico: Este método se puede usar con eficacia en la segmentación de imágenes, en estos métodos, la imagen se modela como un grafo ponderado no dirigido. Por lo general un pixel o grupo de pixeles se asocian con los nodos y los pesos de las aristas definen la similitud entre los pixeles vecinos. La imagen se divide de acuerdo con un criterio de diseño para seleccionar de una manera "muy adecuada" los clústeres. Algunos de estos métodos de particionamiento gráfico más comunes son: cortes normalizados, camino aleatorio, el mínimo corte, particionamiento isoperimétrico y árboles de expansión mínima.
8. Watershed o transformación divisoria: Este método funciona a través de la detección las líneas de agua o líneas divisorias. Una imagen a escala de grises se puede apreciar como un relieve topográfico donde se interpreta el nivel del color gris de un pixel como su altura en el relieve. También el gradiente de una imagen se puede considerar como una superficie topográfica. Los pixeles que tienen intensidades altas de gradientes corresponden a líneas divisorias, que representan los límites de las regiones.

   El agua puesta sobre cualquier pixel encerrado por una línea divisoria común fluye "colina abajo" a un mínimo de intensidad local común. Los píxeles que drenan a un mínimo común forman una cuenca, que representa un segmento de la imagen (un objeto).
9. Segmentación basada en modelos: se basa en la hipótesis fundamental de que las estructuras de interés de una imagen contienen forma geométrica repetitiva. Por tanto, es posible buscar un modelo probabilístico que explica la variación de la forma de la estructura que posteriormente cuando se segmenta una imagen se impone limitaciones para tomar la imagen como modelo elegido a priori.

   Esto implica la selección de los objetos de entrenamiento (ejemplos para probar los modelos), la representación probabilística de la variación de los ejemplos seleccionados y la inferencia estadística entre el modelo y la imagen.

La segmentación basa en el conocimiento implica la forma activa y los modelos de apariencia, contornos activos y una plantilla deformable y métodos basados en niveles.

10. Segmentación multiescala: Las segmentaciones de la imagen se calcula en múltiples escalas y a veces es propagada de gran escala a pequeña escala. Un requisito de este método es que cada región debe estar conectada en algún sitio.

    Dentro de los métodos de segmentación de este tipo podemos encontrar: la segmentación jerárquica de señales unidimensionales, entre otros.
11. Segmentación semiautomática: En este tipo de segmentación semiautomática el usuario define las regiones de interés con un click de ratón y el algoritmo aplica la forma más eficaz para identificar el borde la imagen, dentro de este tipo de método de segmentación encontramos:

    Redes neuronales de segmentación: Se basan en el procesamiento de pequeñas áreas de una imagen utilizando una red neuronal artificial o un conjunto de redes neuronales artificiales. Después de este procedimiento se elabora un mecanismo donde se marca las áreas de una imagen de acuerdo con la categoría reconocida por la red neuronal.

    Un tipo de red neuronal desarrollada para este tipo de aplicación es el mapa de Kohonen, las redes neuronales de parejas de pulsos (PCCNNs) o Pulse-Coupled Neuronal Network son modelos propuestos por corteza visual de un gato y propuesta para un alto rendimiento de procesamiento de imágenes biomiméticas.

    Aproximadamente, desde el año 2000, las PCNNs (redes neuronales de parejas de pulsos) han sido utilizadas en diversas aplicaciones dentro de las cuales tenemos: generación de características, generación de funciones, la extracción de rostros, detección de movimientos, detección de regiones en crecimiento, y reducción de ruido.

    Un PCNNs es una red neuronal de dos dimensiones, cada neurona analiza un pixel de una imagen de entrada, recibiendo la información del color como un estímulo externo, cada neurona se conecta con las neuronas vecinas recibiendo los estímulos locales, los estímulos externos y locales se combinan en un sistema de activación interno, que acumula estímulos hasta exceder el umbral dinámico, dando como resultado una salida de estímulos, a través de procesos iterativos las neuronas PCCNs producen series temporales de impulsos a la salida, esta serie temporal de impulsos contiene información de la imagen de entrada y puede ser usado utilizado para varias aplicaciones de procesamiento de imágenes tales como la segmentación de imágenes o la generación de características, esta técnica comparada con técnicas tradicionales posee varia ventajas importantes incluida la robustez frente al ruido, la independencia de la variación de los

patrones geométricos de entrada, capacidad para pasar por pequeñas variaciones en los patrones de intensidad de entrada, etc.

12. Segmentación basada en color: como se ha explicado anteriormente el color se puede suponer como una unión de tres planos, tres canales, cada uno con sus niveles de intensidad en cada punto o pixel, de una base de color que normalmente es la RGB, para realizar esta segmentación se pueden utilizar algunas de las técnicas o métodos estudiados anteriormente en cada una de las tres capas e integrar estos resultados en la imagen, para que salga la imagen segmentada a color.
13. Segmentación basada en la textura: Este método de segmentación consiste en primero definir modelos de texturas para tratar la imagen como si fuese una función y no una colección de pixeles y poder convertir la ventana de la imagen en un vector de características.

    Los modelos de texturas se pueden dividir en tres clases: modelo basado en estructuras piramidales, modelo definido en campos aleatorios y modelo basado en métodos estadísticos.

    Dependiendo del tipo de estructura a reconocer y el conocimiento a priori de las mismas esta segmentación se puede clasificar en supervisada o no supervisada.

    El objetivo de esta segmentación es identificar un vector de características similares y asignarle una etiqueta.
14. Segmentación basada en conocimiento: consiste en estudiar la imagen resultante de la resta de dos imágenes consecutivas de la secuencia dejando los pixeles que se desplazan con valores distintos de cero al restar las imágenes, mientras que los que no se desplazan se mantienen con valores nulos al realizar la resta. Esta segmentación depende mucho de la iluminación del entorno o ambiente, pues la sustracción de fondo, al restar dos imágenes, se debe tomar un umbral que al momento de binarizar la imagen, sean similares, y esto dependerá básicamente de esta iluminación.

2.5.4.4. Extracción de características

El proceso de extracción de extracción de características consiste en identificar y describir las características de objetos o formas presentes en una imagen como pueden ser color, geometría, textura, superficie, posición, nivel de intensidad, y se pude identificar cada una de ellas o una combinación de varias de ellas.

La extracción de características es muy dependiente de la aplicación que se le quiera dar, existen muchos métodos para la extracción, no obstante, no se cuenta con un método universal ya que se tienen en cuenta los propios requerimientos del problema.

Las características que se extraen de la imagen usando estas técnicas o metodologías deben cumplir las siguientes condiciones:

Capacidad discriminante: resalta las diferentes características para los objetos de diferentes clases.

Fiabilidad: la dispersión de los objetos de una mis clases debe ser pequeña, es decir, deben un amplio rango de detección de estas características sin llegar a tener en cuenta ruidos similares que se lleguen a presentar.

Incorrelación: las características identificadas no deben estar relacionadas entre sí, con el fin, que identifique de manera eficaz características de objetos diferentes.

Rapidez y economía de cálculo: las características deben identificarse en un lapso razonable, con el mínimo de carga computacional requerida.

Invarianza al tamaño, rotaciones y traslaciones: el cambio del tamaño, el ángulo y la posición no deben afectar en la identificación de las características [50].

2.5.4.5. Reconocimiento e interpretación

El reconocimiento es la última fase o etapa de la visión artificial, en esta fase el sistema interpreta finalmente la imagen, clasifica y toma decisiones pertinentes previa programación.

Este proceso debe llevar al sistema de reconocimiento a asignar cada objeto una etiqueta o categoría que comparte alguna característica de interés que las difiere del resto.

El reconocimiento lleva a la clasificación, entendida como la clasificación de diferentes partes del vector de características a grupos o clases y basándose en las características extraídas, llegar al aprendizaje automático, desarrollando técnicas o métodos que le permita a las computadoras aprender y de esta forma reconocer formar u objetos, obtenidas desde la imagen de estudio.

Los algoritmos de aprendizaje se identifican por partir de un conjunto de funciones o aplicaciones discriminantes, con valores en sus parámetros que permiten que se puedan discriminar optimizando las funciones respecto al objeto a reconocer, el proceso termina cuando el conjunto de entrenamiento se clasifica correctamente.

Como se vio anteriormente estos son algunos de los clasificadores que se usan actualmente en la visión artificial:
Geométricos: para esta caracterización de patrones se utilizan el cálculo de distancias, geometría de formas, vectores numéricos, etc.
Estadísticos: se basan en teorías probabilísticas y de estadística, como umbrales, análisis de varianza, covarianzas, dispersión, distribuciones, tendencias, etc.

Igualmente, se pueden obtener diferentes tipos de reconocimiento:
Sintáctico – estructural: Encuentra las relaciones estructurales de los objetos de estudio, utilizando teorías de lenguajes formales, teorías autómatas, etc.
Neuro-reticular: Utiliza redes neuronales que se entrenan para dar respuesta a determinados valores.
Lógico-combinatorio: Se basa en la idea del que modelado del problema debe ser lo más posible cercano a la realidad y de esta forma poder usar conjuntos difusos, lógica simbólica, circuitos combinacionales y secuenciales, etc.

Igualmente, según se tenga un conocimiento previo de datos que permita aprender, la clasificación de estos reconocimientos se puede dar en: supervisada, parcialmente supervisada o no supervisada.

Clasificación supervisada: también conocida como clasificación con aprendizaje, se fundamenta en la disponibilidad en áreas de entrenamiento. Se trata de áreas de las que se conoce a priori la clase a la que pertenecen y que servirán para generar una etiqueta espectral característica de cada una de las clases. Algunos métodos de clasificación supervisada son:

a. Funciones discriminantes: consiste básicamente en: sean dos clases y se busca obtener una función g tal que para un nuevo objeto O, entonces si $g(O) \geq 0$ se asigna a la clase 1 y en otro caso la 2. Si son múltiples clases se busca un conjunto de funciones g_i y el nuevo objeto se ubica en la clase donde la función tome el mayor valor.

b. Vecino más cercano: Un nuevo objeto se ubica en la clase donde este el objeto de la muestra original que más se le parece.

c. Redes neuronales artificiales: Se suponen que imitan a las redes neuronales reales en el desarrollo de tareas de aprendizaje.

Clasificación parcialmente supervisada: También conocida como de aprendizaje parcial. En este caso existe una muestra objetos de solo algunas clases definidas.

Clasificación no supervisada: También conocida como sin aprendizaje. Se utilizan algoritmos de clasificación automática multivariante, en los que los individuos más próximos se van agrupando formando clases.
Así mismo, esta clasificación no supervisada se puede dividir en a priori y a posteriori, según sean clasificadores de un solo paso que utilizan la muestra de aprendizaje para el cálculo de las funciones discriminantes y un cálculo exacto (a priori) o clasificadores con aprendizaje, de procedimientos iterativos de entrenamiento, en el cual, el clasificador aprende a reconocer de forma progresiva los patrones de la muestra de aprendizaje (a posteriori).

Los clasificadores, como base del sistema de reconocimiento, tienen una gran aplicación además de la visión artificial, donde se usan para reconocimiento de caracteres, ya sea caracteres escritos a mano o impresos, reconocimiento facial, reconocimiento de objetos y reconocimiento de huellas dactilares, como son: Previsión meteorológica, reconocimiento de voz, aplicaciones de medicina, interpretación de fotografías de área y de satélite de gran utilidad para propuestas militares o de agricultura, predicción máxima de terremotos, reconocimiento de música, entre otras [5].

2.5.5. Lenguajes de programación y librerías

En la actualidad existen gran cantidad de lenguajes de programación por los cuales podemos realizar el procesamiento de imágenes, no obstante, los que se han utilizado en este trabajo son el lenguaje C++ con las librerías de Open CV y el lenguaje de Python, ya que estos dos lenguajes son los que trabajan de una forma eficaz los hardware de Raspberry Pi.

2.5.5.1. Python

Es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con la naturaleza interpretada, hace

de este lenguaje un aliado eficaz para scripting y desarrollo rápido de aplicaciones en diversas áreas sobre las diversas plataformas.

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, http://www.python.org/, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas, y documentación adicional.

El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables.

Por otro lado, si se compara Python con bibliotecas de C/C++/Java se encuentra que se hace lento el ciclo usual de escribir/compilar/testear/recompilar. Igualmente, si se escribe una batería de pruebas para una de esas bibliotecas, se encuentra que escribir el código de testeo se hace una tarea tediosa. Con Python estos procesos se reducen considerablemente, y se encuentra disponible para sistemas operativo como Windows, Mac OS X y Unix, lo cual, ayuda a realizar las tareas más fácilmente.

Python permite separar un programa en módulos que pueden reusarse en otros programas en Python.

Viene con una gran colección de módulos estándar que se pueden usar como base de otros programas, o como ejemplos para empezar a aprender a programar en Python. Algunos de estos módulos proveen cosas como entrada/salida a archivos, llamadas al sistema, sockets, e incluso interfaces a sistemas de interfaz gráfica de usuario como Tk [57].

Python es un lenguaje interpretado, lo cual puede ahorrar mucho tiempo durante el desarrollo ya que no es necesario compilar ni enlazar. El intérprete puede usarse interactivamente, lo que facilita experimentar con características del lenguaje, escribir programas descartables, o probar funciones cuando se hace desarrollo de programas de abajo hacia arriba. Es también una calculadora de escritorio práctica.

Python permite escribir programas compactos y legibles. Los programas en Python son típicamente más cortos que sus programas equivalentes en C, C++ o Java por varios motivos:

• Los tipos de datos de alto nivel permiten expresar operaciones complejas en una sola instrucción

• La agrupación de instrucciones se hace por sangría en vez de llaves de apertura y cierre.

• No es necesario declarar variables ni argumentos [57].

2.5.5.2. Open CV

OpenCv *(Open Source Computer Vision)* es una librería multiplataforma inicialmente desarrollada por Intel para apoyar a los primeros compiladores Intel C++ y Microsof visual C++ en x86. Es muy utilizada en el procesamiento de imágenes y visión artificial, también se ha utilizado en infinidad de aplicaciones como control de procesos, sistemas de seguridad con detección de movimiento, reconocimiento de objetos, robótica avanzada, etc.

Con base a lo mencionado a la página oficial, es una librería completamente libre y gratuita, pues se distribuye bajo la licencia BSD *(Berkley Software Distribution),* que permite que sea usada libremente en distintos proyectos con propósitos comerciales y de investigación, siempre y cuando, cumpla con las condiciones de la licencia.

Las librerías de OpenCV pueden instalarse tanto bajo Linux como bajo Windows.

OpenCV cuenta con una buena eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. Está escrito en C y C++ optimizados, y se puede ejecutar en GNU/Linux, Windows, Android, iOS y Mac OS X. También es de anotar que hay un activo desarrollo en las interfaces de Python, Ruby, Matlab, y otros lenguajes.

OpenCV se estructura en cinco componentes principales [58], [59]:

- El componente de CV contiene los algoritmos de procesamiento de imágenes y de visión por computador en nivel básico y superior.
- ML es la biblioteca de aprendizaje de máquina, que incluye muchos clasificadores estadísticos y herramientas de agrupación.
- HighGUI contiene rutinas y funciones de I/O (entrada /salida) para el almacenamiento y carga de videos e imágenes.
- CXCore contiene las estructuras básicas y algoritmos, apoyo XML, y funciones gráficas. Recibe información de CV, ML y HighGUI.
- CvAux, que contiene algunos algoritmos experimentales

Con respecto al campo del procesamiento de imágenes y visión artificial, uno de los objetivos de OpenCV es proporcionar una infraestructura accesible y sencilla. La biblioteca de OpenCV contiene un gran número de funciones utilizadas en varios campos de la visión artificial, como es

la inspección de productos, identificación de personas u objetos en movimiento, reconocimiento de rostros humanos en una imagen, imágenes médicas, seguridad, interfaces de usuario, reconstrucción 3D, robótica, etc. [50].

3. DESARROLLO DEL SISTEMA DE DETECCIÓN

En este apartado mencionaremos los pasos secuenciales que se llevaron a cabo para la realización de este trabajo, partiendo desde la selección de los hardware y accesorios necesarios para el desarrollo, pasando por la investigación y análisis de algoritmos y aplicaciones que se tienen relacionados con visión artificial en diferentes campos, siguiendo con la creación del algoritmo adecuado para nuestra aplicación, hasta el montaje para las diferentes pruebas en el ambiente simulado.

3.1 HARDWARE Y PERIFÉRICOS Y/O ACCESORIOS A UTILIZAR

Como uno de los objetivos del trabajo es la aplicación de Hardware y software de bajo coste para la implementación en aplicaciones agroindustriales, procedimos a buscar los hardware que se han venido trabajando en diversos estudios de agricultura de precisión que se pueden encontrar tanto en artículos científicos [5], [6], [30], [45], [60] como en tesis de pregrado y posgrado [5], [50], [53]–[55], [61], [62].

Dentro de los hardware que se usaron tenemos:

3.1.1 Computador Portátil

El ordenador fue un modelo portátil con webcam incorporada para hacer pruebas preliminares de los algoritmos, es marca HP Pavilion x360, procesador Intel® Core i5-7200 CPU@ de séptima generación 2.5 GHz – 2.71 GHz, RAM de 8.00 GB y disco duro de 1 TB.

3.1.2 Tarjeta embebida Raspberry Pi 2

Para la adquisición y procesamiento de las imágenes se utilizará Raspberry Pi 2 junto con el módulo oficial de cámara oficial de Rapsberry Pi y una Webcam comercial [54].

La Raspberry Pi es un controlador de placa reducida de bajo coste, desarrollado en el Reino Unido con el fin de facilitar el acceso a la tecnología de computación a un público más amplio y dar herramientas lúdicas en la enseñanza de esta ciencia [63].

Existen otros dispositivos similares a las Raspberry Pi como las tarjetas Arduino, no obstante, estas tienen características muy inferiores a las Raspberry Pi.

Para el proyecto se usó una tarjeta Raspberry Pi 2 (figura 41), debido a su practicidad, rendimiento y precio respecto a otros hardware similares, en la figura 42, se presenta una tabla comparativa de los hardware de desarrollo similares a la Raspberry Pi [53].

Figura 41. Raspberry Pi 2 Modelo B

(Fuente: [53])

	Raspberry Pi B	OLinuXino A20	Cubieboard2	ODROID XU3	Beagle Board
Frecuencia procesador	700/1000 MHz	1000 MHz	1000 MHz	2000 MHz	1000 MHz
Nº núcleos	1	2	2	8	1
RAM	512 MB	512 MB	1024 MB	2048 MB	512 MB
Nº USB	2	2	2	5	1
Ethernet	100 Mbit	100 Mbit	100 Mbit	100 Mbit	100 Mbit
Wi-Fi Bluetooth	No/No	No/No	No/No	No/No	No/No
SATA	No	Sí	Sí	No	No
GPIO	26	160	67	30	96
Cámara	Sí (5 Mp)	No	Sí (0,3 Mp)	Sí (1 Mp)	Sí (1 Mp)
Precio	27,40 €	65 €	62,41 €	144 €	100 €

Figura 42. Comparación de Raspberry Pi con la competencia

(Fuente: Tomada de la Memoria del trabajo de fin de Máster “Estudio de viabilidad de un sistema basado en Raspberry Pi para aplicaciones de Inspección industrial por Visión Artificial”, David Silva Montemayor, pág. 32 [51])

Algunas de las características técnicas de esta placa son:

- SoC: Broadcom BCM2837 (CPU + GPU + DSP + SDRAM + puerto USB). · CPU: ARM Cortex-A53 a 1.2 GHz, 64-bit, 4 núcleos. · Juego de instrucciones: RISC de 32 bits.
- GPU: Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia), 1080p30 H.264/MPEG-4 AVC.
- Memoria (SDRAM): 1 GB compartido con la GPU. · Puertos USB 2.0: 4
- Entradas de vídeo: Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la Raspberry Pi Foundation (RPF).
- Salidas de vídeo: Conector RCA (PAL y NTSC), HDMI (rev 1.3 y 1.4), Interfaz DSI para panel LCD.
- Salidas de audio: Conector de 3.5 mm, HDMI.
- Almacenamiento integrado: microSD.
- Conectividad de red: 10/100 Ethernet RJ-45, WiFi 801.11n y Bluetooth 4.1
- Periféricos de bajo nivel: 17 x GPIO, SPI, I2C, UART
- Consumo energético: 800 mA (4W)
- Fuente de alimentación: 5 V vía Micro USB o GPIO Header.
- Dimensiones: 85.6 mm x 53.98 mm.
- Sistemas operativos soportados: GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux. RISC OS. Windows 10 IoT Core [54].

3.1.3 Memoria Micro SD

Para correr el sistema operativo con el cual trabaja la Raspberry Pi, se debe tener una memoria micro SD, la cual, hospedará y ejecutará dicho sistema.

La tarjeta micro SD que se utilizó para la etapa del desarrollo experimental del trabajo y que cumplió básicamente dos funciones como fueron: alojar el sistema operativo Raspbian del sistema embebido y almacenar las diferentes imágenes, videos y programas del proyecto, fue la tarjeta micro SD Kingston de 16 GB, de clase 10 que soporta información superior, videos full HD, videos 3D y videos full HD con calidad de Cine (1080p), compatible con interfaz de UHS-I.

En la figura 43, se muestra la memoria usada en el proyecto.

Figura 43. Tarjeta Micro SD usada en el desarrollo

(Fuente: Propia)

3.1.4 Cargador Portátil

Como la Raspberry debe quedar alimentada de forma inalámbrica, se usará un cargador de teléfono móvil marca ADATA, Modelo PV120 con capacidad de 5100mAh de 2.1 A a 5 v DC, conectado a través del puerto micro USB que tiene la tarjeta, este cargador se puede observar en la figura 44.

Figura 44. Cargador Portátil de dispositivo móvil

(Fuente: Propia)

3.1.5 Dongle Wifi Raspberry

Dentro de la funcionalidad del sistema, se debe tener una conexión inalámbrica debido a que este sistema irá directamente acoplado al Drone, se usó una conexión inalámbrica por medio de Wifi, por ende, se hizo necesario utilizar un dongle para wifi, marca EDUP, modelo: EP-N8508GS, interfaz: USB, Normas: IEEE 802.11b / g / n, velocidad de 150Mbps, seguridad: WPA / WPA2 / WEP, este elemento se muestra en la figura 45.

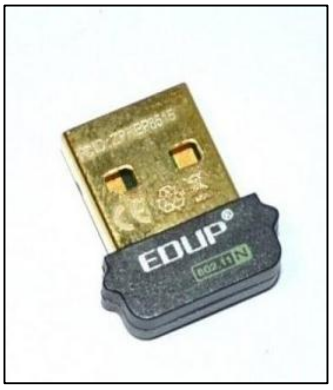

Figura 45. Dongle Wifi Raspberry Pi

(Fuente: dualtronica.com)

3.1.6 Webcam Genius como elemento para la adquisición de imágenes

En este trabajo usaremos una webcam marca Genius FaceCam 1000X, la cual, cuenta con las siguientes características técnicas:

- Sensor de imagen CMOS de píxeles de alta definición 720p
- tipo de lente objetivo de enfoque manual
- Formato de archivo AVI/WMV
- Resolución (DPI)1MP, 1280 x 720, 640 x 480 pixels
- Peso50 g
- Dimensiones (A x A x P)20 x 22 x 60 mm

En la figura 46. Se muestra una fotografía de la webcam comercial usada en el trabajo

Figura 46. Webcam comercial

(Fuente: http://cl.geniusnet.com/product/facecam-1000x)

Otras de las opciones disponibles para realizar la adquisición de imágenes para estos proyectos es la Raspicam, como se dijo anteriormente, este hardware es el oficial de la Raspberry Pi, una de las desventajas de este dispositivo es que no tiene carcasa de protección, por tanto, hay que diseñarla para la aplicación que se requiera, no obstante, su conexión que es a través de un puerto CSI, lo cual, permite dejar más puertos USB disponibles y su resolución, la cual, es de 5 megapixeles, adicionalmente, la capacidad de tomar video a 1080p (pixeles) y 30 (frames per second) 720p y 60 fps o a 640x480p y 60 o 90 fps, lo hace una muy buena alternativa para el desarrollo de estos prototipos.

En la figura 47, se muestra la Raspicam usada en el prototipo del proyecto

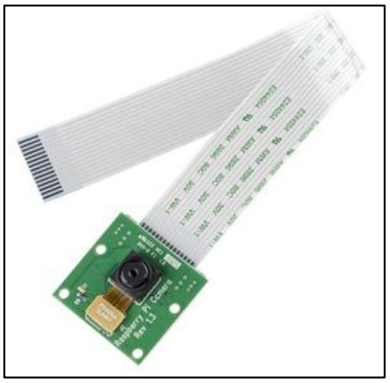

Figura 47. Raspicam

(Fuente: Propia)

Así mismo, su flexibilidad en la conexión y reducido tamaño son otras de sus grandes ventajas para implementación en proyectos, donde se requiere que el prototipo de adquisición de imágenes sea pequeño y liviano.

3.1.7 Luxómetro HS1010A

Como se puedo evidenciar en los estudios que se han citado constantemente, una de las variables a tener en cuenta para la aplicación de un sistema de visión artificial es la iluminación, ya que la generación de sobras o ruidos se producen a consecuencia de una iluminación no uniforme, sin embargo, nuestro estudio consiste en verificar si un sistema de visión artificial para la detección de enfermedades dentro del lulo utilizando hardware y software de bajo coste es viable o no , para contribuir en la implementación de nuevas tecnologías en estos campos del sector productivo de nuestro país.

En este sentido, fue necesario contar con este dispositivo, un luxómetro HS1010A, el cual, es un elemento que nos permite medir la cantidad de flujo luminoso sobre un área en específica.

Las especificaciones técnicas del luxómetro son:

- Repetibilidad: $\pm$2%
- Tamaño (largo x ancho x alto): 108 x 60 x 32 mm
- Característica de la temperatura: $\pm$0.1%/°C
- Resolución: 1 LUX x 0.1 FC
- Fotodetector: Un fotodiodo de silicio con filtro
- Rango de medición: 1 – 200000 LUX
- Velocidad de medición: 2 veces por segundo
- Precisión de medición: $\pm$4% $\pm$ 10 dgts (<10000LUX)

En la figura 48, se presenta una imagen del dispositivo que se usó.

Figura 48. Luxómetro HS1010A Usado para la validación del algoritmo de visión artificial

(Fuente: http://vi.vipr.ebaydesc.com/ws/eBayISAPI.dll?ViewItemDescV4&item=282881594847&t=0&tid=7710&category=42632&seller=clever-deals&vipguid=3f08c9871620ab15af3476bbff9ed2e)

3.1.8 Cámara fotográfica Canon EOS Rebelt T3 EOS 1100

Debido a que se quiere colocar en práctica el algoritmo de visión artificial en el campo real del cultivo del lulo y ya que por cuestiones logísticas del traslado no se nos es posible estar constantemente en el cultivo, procedimos a usar una cámara profesional Canon Rebel T3 (figura 49), para captura las imágenes de la enfermedad del lulo que se quiere identificar con la visión por computador, estas imágenes fueron usadas para la verificación constante del algoritmo y para las pruebas finales del programa en el entorno controlado de iluminación, como era importante que la calidad de las imágenes fueran lo más parecido posible a lo encontrado en la realidad, se usó una cámara profesional de muy buenas características (tabla 2):

Figura 49. Cámara fotográfica profesional usada en el proyecto

(Fuente: https://www.decamaras.com/imagen/camaras11/canon-1100d.jpg)

Tabla 2. Tabla 2. Especificaciones de la cámara fotográfica profesional EOS Rebel T3

Sensor	
Tipo	CMOS
Factor de recorte	1,6 X
Megapixeles	12,2 MP
Sensibilidad a la luz	ISO 6.400
Sensibilidad a la luz (incremento)	Desconocido
Limpieza del sensor	No
Sensor avanzado	
Verdadera resolución	12,2 MP
Resolución nativa	4272 × 2848
Tamaño de píxel	26,8 µm²
Pantalla	
Tipo	LCD
Tamaño	6,9cm
Resolución	230 mil puntos
Pantalla táctil	No
Voltea a cabo	No
Vista en vivo	Sí
Objetivos	
Lentes disponibles	220 lentes
Montura del objetivo	Canon EF
Factor de forma	
Tamaño	130 × 100 × 78 mm
Profundidad	7,8cm
Peso	459 g
Lentes intercambiables	Sí

Resistente al agua	No
Sellado para el clima	No
Motor de enfoque incorporado	No
Visor	
Tipo	Pentaespejo
Tamaño del visor	0,54 X
Cobertura	95 %
Cine	
Formato	720p @ 30 fps
Soporta 24p	No
Frecuencia de fotogramas de alta velocidad	Ninguno
Jack de micrófono externo	No
Enfoque automático	Ninguno
Enfoque continuo	n/a
Todos los formatos	720p @ 30 fps
Características	
Panorama	No
HDR (Alto Rango Dinámico)	No
3D	No
Estabilización de imagen	Ninguno
Soporta RAW	Sí
GPS	No
Sistema de enfoque	
Enfoque automático	Detección de fase
Puntos de enfoque	9
Puntos de enfoque de tipo cruz	1
Almacenamiento	
Ranuras de almacenamiento	1
Formatos compatibles	SD
	SDHC
	SDXC
Calificación de DXO Mark	
La calidad de imagen	62
Profundidad del color	21,9 bits
Rango dinámico	11 EV
Rendimiento con poca luz	ISO 755

(Fuente: http://snapsort.com/es/comparar/Canon-EOS-Rebel-T5-vs-Canon-T3/especificaciones)

3.2 Instalación del sistema operativo y puesta en marcha de la Raspberry Pi 2

Posteriormente, ya adquirido todos los equipos y hardware necesarios para el desarrollo del proyecto, se inicia con el proceso de puesta en funcionamiento del sistema embebido a utilizar, por ende, uno de los pasos importantes a realizar es la instalación del sistema operativo en la tarjeta Raspberry Pi, que para nuestro caso se instaló el sistema operativo Raspbian, aunque en realidad al ser un open Hardware, se podría haber utilizado otro sistema operativo dispuesto actualmente en internet como: Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IOT Core, OSMC, Librelec, Risc OS, entre otros.
Como se ha mencionado anteriormente, este tipo de tarjetas embebidas requieren de una tarjeta micro SD, en donde, se va a alojar el sistema operativo completo, para su correcto funcionamiento al momento del desarrollo de las aplicaciones que se estén corriendo o usando.

Inicialmente se formatea la memoria micro SD a través de un software recomendado llamado SD Formatter, el cual, se encuentra gratuito en internet. El proceso de formateo es sencillo, simplemente se inserta la tarjeta micro SD al PC y en el programa se busca en la opción Drive la ruta de la memoria micro SD.

Una vez escogida la ruta se procede a darle click al botón format y la tarjeta quedará formateada.

Posteriormente, se procede a subir la imagen del sistema operativo de Raspbian a la tarjeta, para esto, se usa la aplicación win32-disk-imager, y cuando se abra esta aplicación como se observa en la figura 50, se coloca la ruta donde se haya descargado el sistema operativo y la ubicación donde se quiera colocar la imagen, que para este caso será la tarjeta micro SD.

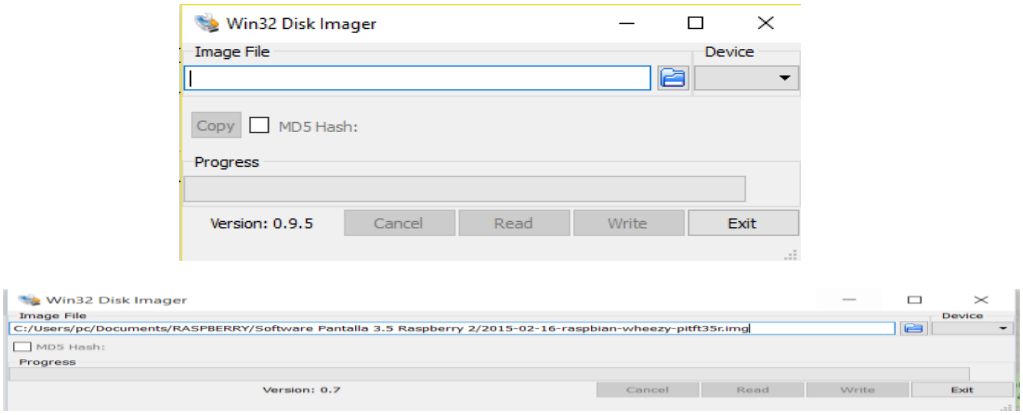

Figura 50. Carga de la imagen Raspbian a la SD con el software Win 32

(Fuente: Propia)

Una vez ya instalado el sistema operativo en nuestra micro e insertada en la Raspberry PI, procederemos a hacer la respectiva configuración.

Cuando se encienda la Raspberry PI, se comienza a ejecutar una serie de códigos. Para trabajar con la Raspberry PI, se puede hacer de dos formas mediante líneas de código o a través de una interfaz de usuario.

En la figura 51, se puede observar algunas de las etapas de configuración del sistema operativo

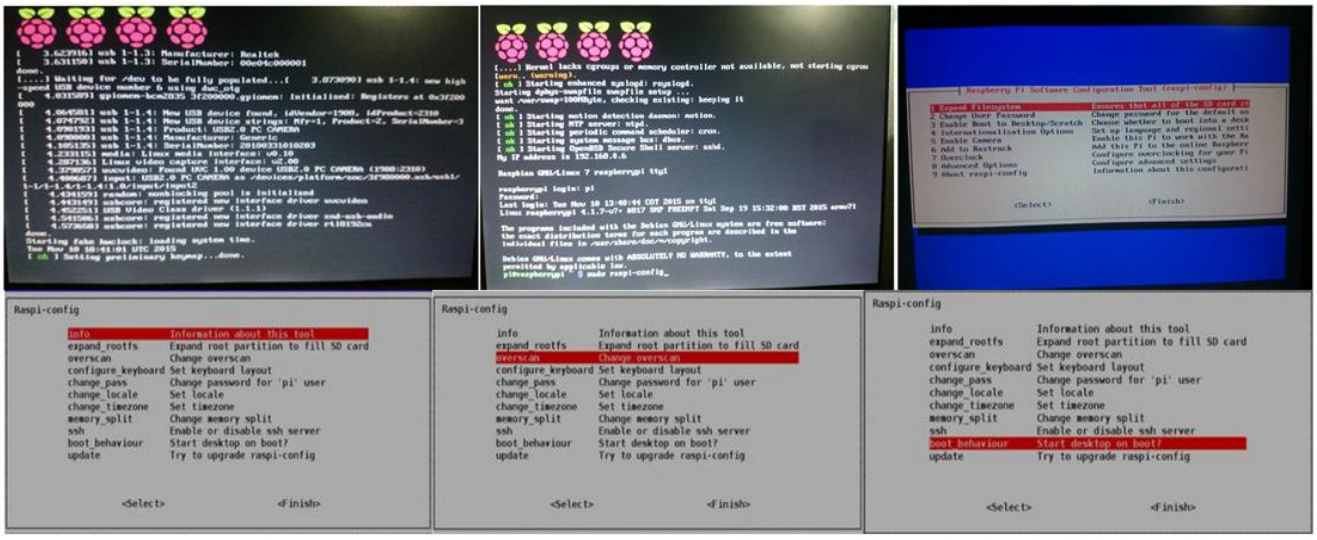

Figura 51. Etapas de configuración del S.O Raspbian en la Raspberry Pi

(Fuente: Propia)

3.3 Implementación, puesta en funcionamiento y verificación de aplicaciones o programas similares encontrados en la literatura y la web

Ya puesta a punto la tarjeta se procedió a realizar una serie de pruebas de funcionamiento del hardware, para esto iniciamos la búsqueda de algunos algoritmos que hicieran procesos similares a los que íbamos a requerir para nuestro trabajo, en este sentido, iniciamos con el habitual algoritmo de detección o reconocimiento de rostros, o como otras personas lo llaman reconocimiento facial, después evaluamos otro algoritmo de mucho interés para nuestro estudio como es la detección o reconocimiento de patrones a través de rutinas orientadas al cálculo geométrico, como se evidencia en el algoritmo de la identificación de la apertura de la mano, seguidamente analizamos otro algoritmo fundamental en el estudio, ya que una característica importante de la enfermedad que se quiere identificar tiene un color característico, este algoritmo

es el de la detección de colores o identificación de colores, y por último, también validamos otro algoritmo de interés en la identificación de figuras geométricas, que es el de la detección de figuras circulares, el cual, es de vital importancia para el proyecto, ya que según la revisión de la literatura que se hizo para la enfermedad de la mancha de la “Alternaria” una de sus características es la forma circular de sus manchas. Cada uno de estos algoritmos serán evaluados en ambientes controlados de iluminación y distancia, usando objetos reales o impresos a excelente calidad, para su reconocimiento e identificación por parte del algoritmo.

3.3.1 Detección Facial

La revolución tecnológica actual y el interés multidisciplinar en desarrollar nuevas metodologías y productos relacionados al reconocimiento de patrones en diferentes aplicaciones, dentro de las que tenemos el reconocimiento facial o de rostros [64], ha permitido que muchos fabricantes de dispositivos electrónicos o hardware, implementen este tipo de algoritmos tanto para uso de entretenimiento como se ven en algunas aplicaciones ya desarrolladas y publicadas en sitios web, como para aplicaciones de seguridad y salud entre otras, en tal sentido, se procedió a realizar una búsqueda de estos algoritmos e identificar los más usados y adecuarlos, con el fin, de validarlos, a través de una serie de pruebas o ensayos en ambientes controlados para determinar de cierta forma, su eficacia y eficiencia en ciertos ambientes de aplicación.

El algoritmo de reconocimiento facial que se escogió para su análisis y validación es el siguiente tomado de [65]:

Para la fase de reconocimiento se trabaja el siguiente código, mostrado en la figura 52.

```
import cv2, sys, numpy, os
size = 4
fn_haar = 'haarcascade_frontalface_alt.xml'
fn_dir = 'att_faces/orl_faces'
# Part 1: Creando fisherRecognizer
print('Formando...')
# Crear una lista de imagenes y una lista de nombres correspondientes
(images, lables, names, id) = ([], [], {}, 0)
for (subdirs, dirs, files) in os.walk(fn_dir):
    for subdir in dirs:
        names[id] = subdir
        subjectpath = os.path.join(fn_dir, subdir)
        for filename in os.listdir(subjectpath):
            path = subjectpath + '/' + filename
            lable = id
            images.append(cv2.imread(path, 0))
            lables.append(int(lable))
        id += 1
(im_width, im_height) = (112, 92)
# Crear una matriz Numpy de las dos listas anteriores
(images, lables) = [numpy.array(lis) for lis in [images, lables]]
# OpenCV entrena un modelo a partir de las imagenes
model = cv2.createFisherFaceRecognizer()
model.train(images, lables)
# Part 2: Utilizar fisherRecognizer en funcionamiento la camara
haar_cascade = cv2.CascadeClassifier(fn_haar)
webcam = cv2.VideoCapture(0)
while True:
    (rval, frame) = webcam.read()
    frame=cv2.flip(frame,1,0)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    mini = cv2.resize(gray, (gray.shape[1] / size, gray.shape[0] / size))
    faces = haar_cascade.detectMultiScale(mini)
    for i in range(len(faces)):
        face_i = faces[i]
        (x, y, w, h) = [v * size for v in face_i]
        face = gray[y:y + h, x:x + w]
        face_resize = cv2.resize(face, (im_width, im_height))
        # Intentado reconocer la cara
        prediction = model.predict(face_resize)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)
        # Escribiendo el nombre de la cara reconocida
        # [1]
        if prediction[1]<500:
         cv2.putText(frame,
          '%s - %.0f' % (names[prediction[0]],prediction[1]),
          (x-10, y-10), cv2.FONT_HERSHEY_PLAIN,1,(0, 255, 0))
   # La variable cara tendrá el nombre de la persona reconocida
   cara = '%s' % (names[prediction[0]])
    cv2.imshow('OpenCV', frame)
    key = cv2.waitKey(10)
    if key == 27:
        break
```

Figura 52. Código en Python para el reconocimiento facial

(Fuente: [63])

Sin embargo, antes de que el algoritmo comience a detectar, se le debe entrenar con el rostro que va a etiquetar, para esto, se debe ejecutar el algoritmo de la fase de captura, el cual, se realiza implementado el código de la figura 53.

```
import cv2, sys, numpy, os
size = 4
fn_haar = 'haarcascade_frontalface_alt.xml'
fn_dir = 'att_faces/orl_faces'
fn_name = sys.argv[1]
path = os.path.join(fn_dir, fn_name)
if not os.path.isdir(path):
    os.mkdir(path)
(im_width, im_height) = (112, 92)
haar_cascade = cv2.CascadeClassifier(fn_haar)
webcam = cv2.VideoCapture(0)

count = 0
while count < 20:
    (rval, im) = webcam.read()
    im = cv2.flip(im, 1, 0)
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    mini = cv2.resize(gray, (gray.shape[1] / size, gray.shape[0] / size))
    faces = haar_cascade.detectMultiScale(mini)
    faces = sorted(faces, key=lambda x: x[3])
    if faces:
        face_i = faces[0]
        (x, y, w, h) = [v * size for v in face_i]
        face = gray[y:y + h, x:x + w]
        face_resize = cv2.resize(face, (im_width, im_height))
        pin=sorted([int(n[:n.find('.')]) for n in os.listdir(path)
               if n[0]!='.' ]+[0])[-1] + 1
        cv2.imwrite('%s/%s.png' % (path, pin), face_resize)
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)
        cv2.putText(im,fn_name, (x - 10, y - 10), cv2.FONT_HERSHEY_PLAIN,
            1,(0, 255, 0))
        count += 1
    cv2.imshow('OpenCV', im)
    key = cv2.waitKey(10)
```

Figura 53. Código de captura del programa de reconocimiento facial

(Fuente: [63])

A continuación, explicaremos cada uno de los comandos más relevantes de este algoritmo.

- cv2.createFisherFaceRecognizer(): Más que un comando es un algoritmo que establece y obtiene fácilmente todos los elementos internos del modelo, además, posee un constructor virtual, ya que permite extraer elementos existentes de otros algoritmos similares ya publicados y/o registrados, contiene una configuración de recuperación de parámetros a través de cadenas de texto, además puede leer y escribir los parámetros en archivos XML o YAML [66], [67]. Cada derivado del algoritmo puede almacenar todos sus parámetros y luego leerlos nuevamente, no hay necesidad de implementarlo nuevamente.
- cv2.CascadeClassifier(fn_haar): Es un comando de cascada de clasificadores potenciados que trabajan con características similares a haar, y que se entrenan con unos cientos de vistas de muestra de un objeto particular llamados ejemplos positivos, que se escalan al mismo tamaño (ejemplo 20x20) y ejemplos negativos: imágenes arbitrarias del mismo tamaño.

- cv2.VideoCapture(0): lee una imagen de un bufer en la memoria.
- Model.predict (face resize): Este comando predice o identifica una etiqueta y la relaciona con el objeto o valor indentificado.
- cv2.rectangle: permite imprimir un rectángulo en el objeto detectado
- haar_cascade.detectMultiScale: es un clasificador en cascada potenciados que trabajan con características similares a haar y Cumple la función de detectar objetos de diferentes tamaños en la imagen de entrada y devuelve los objetos detectados como una lista de rectángulos [66], [67].

En la figura 54, se muestra el trabajo experimental en donde se modificaba la distancia de detección y se tomaba la data tanto de la distancia, como de la iluminancia y el porcentaje de detección o rata de aciertos del algoritmo [68].

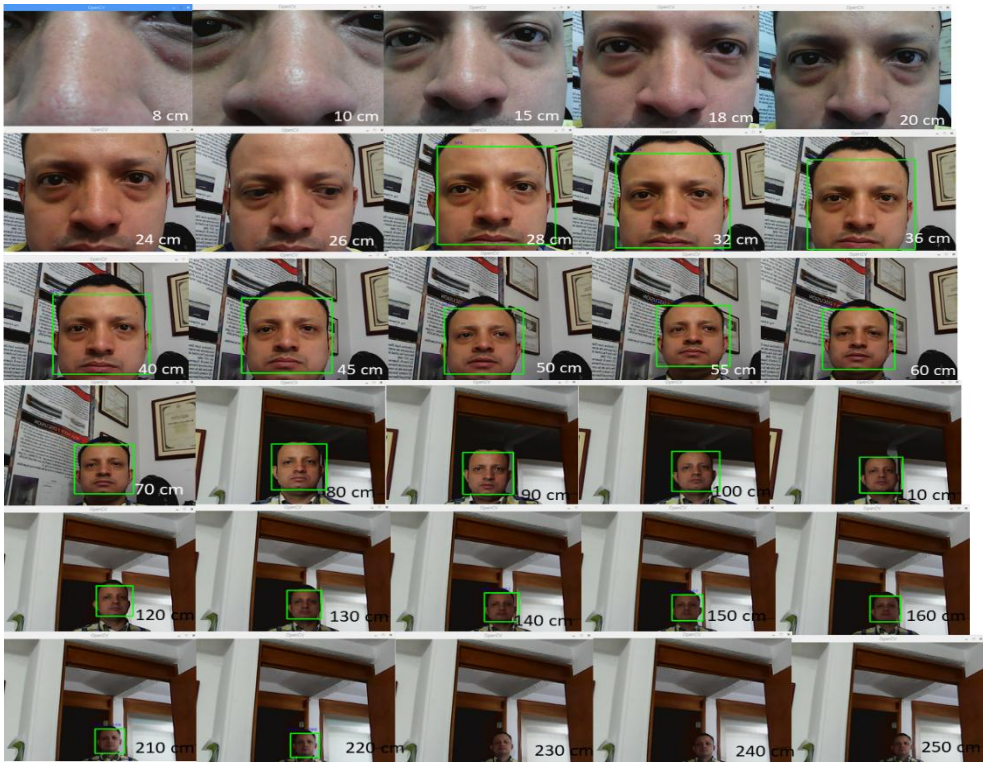

Figura 54. Trabajo experimental realizado al algoritmo de reconocimiento facial aplicando clasificadores en cascada tipo Haar

(Fuente: Propia)

Los resultados del porcentaje de detección en relación con la distancia e iluminancia del algoritmo desarrollado se pueden ver en la figura 55.

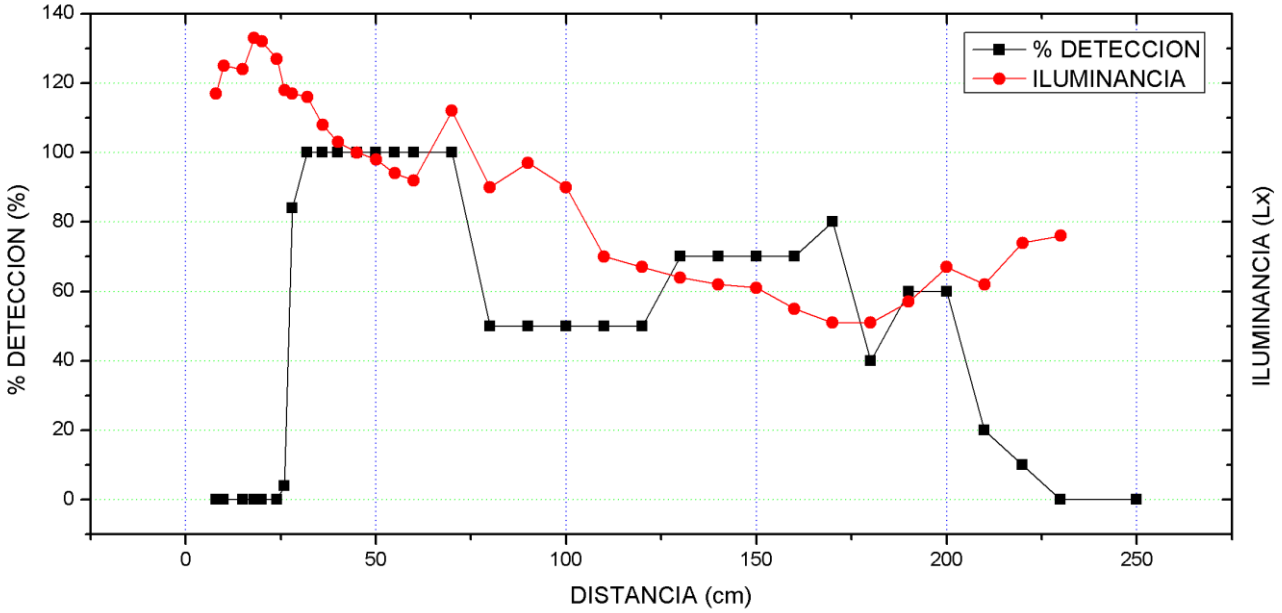

Figura 55. Resultados del trabajo experimental del algoritmo de reconocimiento facial con clasificadores cascada tipo Haar

Fuente: (Propia)

3.3.2 Detección de mano abierta

Otros de los algoritmos analizados de nuestro interés son los de reconocimiento de patrones a través de rutinas orientadas al cálculo geométrico, estos patrones se fundamentan primero en el filtrado y segmentación de la imagen para identificar el contorno y puntos de interés de ubicación, posteriormente estos puntos son etiquetados y finalmente se procede a realizar una serie de cálculos matemáticos como la envolvente convexa que permite determinar figuras geométricas establecidas como círculos, triángulos, cuadrados, etc [68].

En la figura 56, muestra los defectos de convexidad del contorno de la mano

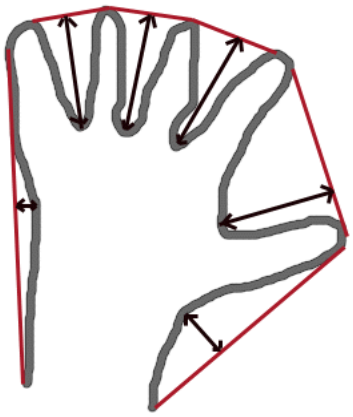

Figura 56. Defectos de convexidad del contorno de la mano

(Fuente: https://docs.opencv.org/3.1.0/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a)

Para este algoritmo (figura 58), donde el objetivo es identificar si la mano está abierta o cerrada sin usar ninguna herramienta de entrenamiento o clasificadores, se procedió a realizar cálculos de geometría entre los puntos de interés y de acuerdo con estos resultados se podía determinar si la mano estaba abierta o cerrada.

Para el desarrollo de este código se usaron los siguientes comandos de la librería de open CV:

- cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY: Este comando convierte la imagen original en una imagen a escala de grises.
- cv2.GaussianBlur(grey, value, 0): Este comando desenfoca la imagen usando un filtro gaussiano que puede ser pasa-bajas denotado por la siguiente función de transferencia:

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

- cv2.threshold: Este comando permite utilizar diferentes tipos de umbralización en las imágenes de trabajo, este comando básicamente realiza la siguiente operación, si el valor del pixel es mayor que un valor de umbral, se le asigna un valor determinado, OpenCv aporta diferentes tipos de umbralización que son:

  cv2.THRESH_BINARY

cv2.THRESH_BINARY_INV

cv2.THRESH_TRUNC

cv2.THRESH_TOZERO

cv2.THRESH_TOZERO_INV

En la figura 57, se presenta las gráficas de los diferentes tipos de umbralización disponibles en OpenCV.

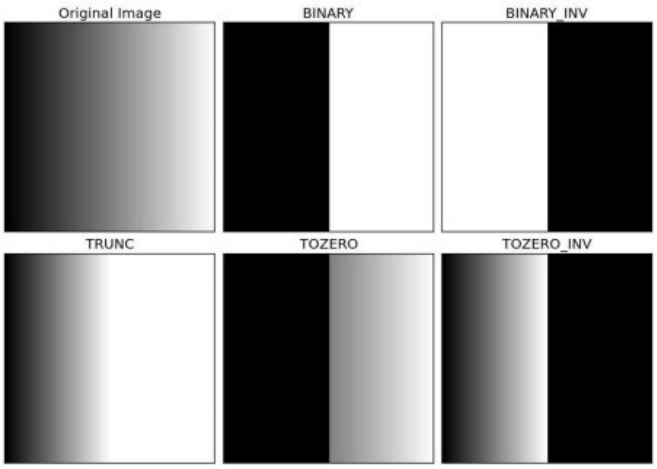

Figura 57. Tipos de Umbralización aportados por OpenCV

(Fuente:https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html)

- cv2.findContours: Este comando permite encontrar los contornos en una imagen binaria, estos contornos son muy útiles al momento de identificar formas y la detección y reconocimiento de objetos.
- cv2.contourArea(cnt): La función calcula el área de un contorno. De manera similar a los momentos, el área se calcula utilizando la fórmula Verde. Por lo tanto, se calcula el área con base a la cantidad de pixeles con valores distintos a cero.
- cv2.boundingRect(cnt): Este comando implementa y calcula un rectángulo delimitador del contorno de un conjunto de pixeles identificados desde la parte superior derecha a un punto determinado.
- cv2.convexHull(cnt): Este comando encuentra el casco convexo de un conjunto de puntos.

Se define entonces que la envolvente convexa de un conjunto de puntos x de dimensión n como la intersección de todos los conjuntos convexos que tienen a X.

Dados k puntos x_1, $x_{2, \ldots,}$ x_k su envolvente convexa C viene dada por la expresión:

$$C(X) = \left\{ \sum_{i=1}^{k} \alpha_i x_i \,\middle|\, x_i \in X, \alpha_i \in R, \alpha_i \geq 0, \sum_{i=1}^{k} \alpha_i = 1 \right\}$$

- np.zeros(crop_img.shape,np.uint8): asigna una matriz de ceros a la imagen de trabajo.
- cv2.drawContours: el comando permite delimitar el contorno a través de líneas o rellena el área del contorno.

- cv2.convexityDefects (cnt,hull): ya vimos en comandos anteriores lo que se define como el casco convexo de un conjunto de puntos y la limitación del mismo, en tal sentido, cualquier desviación de un punto, línea u objeto de este casco se puede considerar como defecto de convexidad, para esta aplicación OpenCV cuenta con este comando, que permite encontrar estos defectos de convexidad.

- math.sqrt: Este comando permite realizar operaciones matemáticas, en este caso se realiza el cálculo de una raíz cuadrada.

- cv2.circle: este comando dibuja círculos en la imagen de trabajo.

- cv2.pointPolygonTest(cnt,far,True): El comando determina si un punto está dentro de un contorno, afuera o se encuentra sobre un borde (o coincide con un vértice). Devuelve un valor positivo (adentro), negativo (afuera) o cero (en un borde), correspondientemente.

- cv2.line(crop_img,start,end,[0,255,0],2): Este comando dibuja el segmento de una línea que conecta dos puntos.

```
import cv2
import numpy as np
import math
cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, img = cap.read()
    cv2.rectangle(img,(300,300),(100,100),(0,255,0),0)
    crop_img = img[100:300, 100:300]
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    value = (37,37)
    blurred = cv2.GaussianBlur(grey, value, 0)
    _, thresh1 = cv2.threshold(blurred, 127, 255,
                               cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    #cv2.imshow('Thresholded', thresh1)
    #print cv2.findContours(thresh1.copy(),cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)
    _, contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)

    max_area = -1
    print len(contours)
    for i in range(len(contours)):
        cnt=contours[i]
        area = cv2.contourArea(cnt)
        if(area>max_area):
            max_area=area
            ci=i
    cnt=contours[ci]
    x,y,w,h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)
    hull = cv2.convexHull(cnt)
    drawing = np.zeros(crop_img.shape,np.uint8)
    cv2.drawContours(drawing,[cnt],0,(0,255,0),0)
    cv2.drawContours(drawing,[hull],0,(0,0,255),0)
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)
    count_defects = 0
    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])
        a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
        b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
        c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
        if angle <= 90:
            count_defects += 1
            cv2.circle(crop_img,far,1,[0,0,255],-1)
        dist = cv2.pointPolygonTest(cnt,far,True)
        cv2.line(crop_img,start,end,[0,255,0],2)
        cv2.circle(crop_img,far,5,[0,0,255],-1)
    if count_defects>3:
        cv2.putText(img,"ABIERTA", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 4)
    else:
        cv2.putText(img,"CERRADA", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 4)
    cv2.imshow('drawing', drawing)
    cv2.imshow('end', crop_img)
    cv2.imshow('Gesture', img)
    all_img = np.hstack((drawing, crop_img))
    cv2.imshow('Contours', all_img)
     k = cv2.waitKey(10)
     if k == 27:
          cap.release()
          break
```

Figura 58. Código para identificar apertura de la mano usando reconocimiento de patrones a través de rutinas orientadas al cálculo geométrico

(Fuente: https://docs.opencv.org/3.1.0/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a))

Igualmente, se procedió a realizar la evaluación del funcionamiento de este algoritmo, donde se le variaba la distancia del objeto a identificar con un valor de iluminancia no constante, en la figura 59 se puede apreciar esta evaluación.

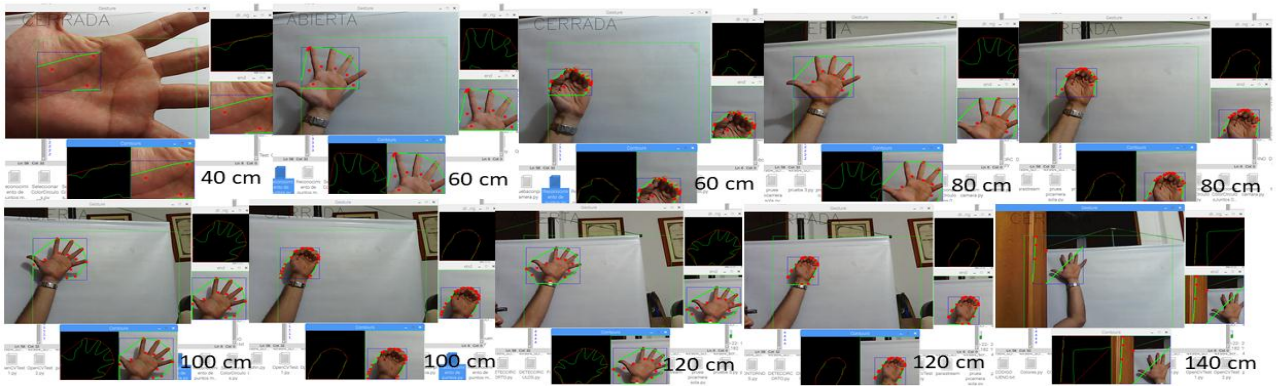

Figura 59. Validación del algoritmo de rutinas orientadas al cálculo geométrico

(Fuente: Propia)

En la figura 60, se puede apreciar el comportamiento de la detección del objeto de este código, usando esta estrategia o método de reconocimiento de patrones.

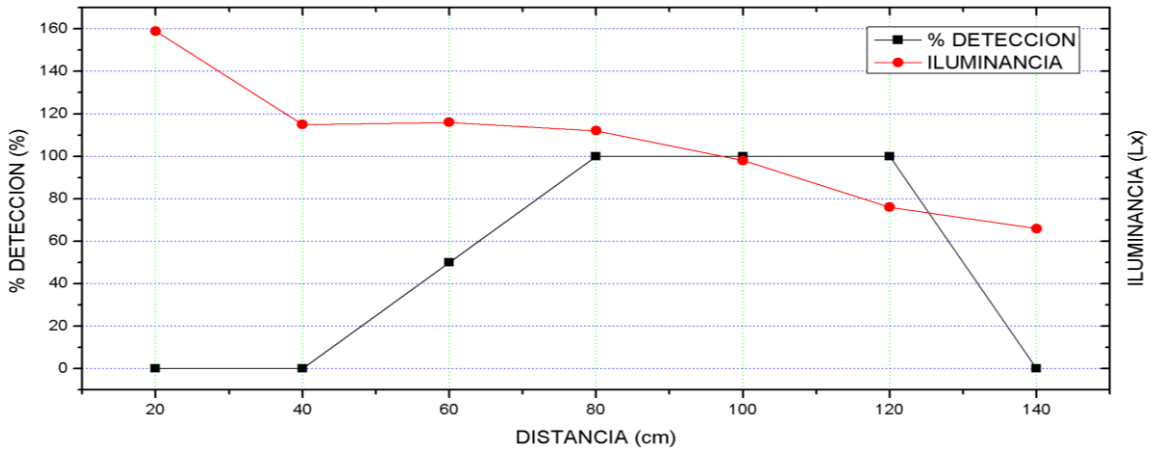

Figura 60. Resultados del comportamiento de detección del algoritmo orientado a rutinas de cálculos geométricos

Fuente propia

3.3.3 Detección de Colores

La detección de colores se puede mencionar como una estrategia interesante al momento de segmentar o reconocer tanto patrones como objetos, a continuación se presenta un algoritmo de segmentación o clasificación a través de colores, este tipo de algoritmos trabajan con algunos filtros y máscaras que permite el reconocimiento más exacto del color que se quiere detectar, primero se hace una selección del color a identificar, seleccionando de una imagen base el color que se quiere sea detectado en la imagen, posteriormente, al agregarle esta información se le puede modificar el rango de máximo y mínimo de los valores del color en la escala RGB.

Para realizar este algoritmo que se muestra en la figura 61, se usaron los siguientes comandos:

- cv2.EVENT_LBUTTONDOWN, cv2.EVENT_MOUSEMOVE: Estos comando permite generar alguna acción dentro de la imagen de trabajo mientras se activa algunas funcionalidades del mouse como puede ser click derecho (oprimido o no), click izquierdo (oprimido o no) y la traslación del mismo a cualquier coordenada en la imagen.
- cv2.inRange(frame,rangomin,rangomax): Este comando permite extraer los objetos de un color determinado aplicando un umbral en la imagen de trabajo.

- cv2.morphologyEx(mascara,cv2.MORPH_OPEN,kernel): Las transformaciones morfológicas son algunas operaciones simples basadas en la forma de la imagen. Normalmente se realiza en imágenes binarias. Necesita dos entradas, una es nuestra imagen original, la segunda se llama elemento estructurador o kernel que decide la naturaleza de la operación. Dos operadores morfológicos básicas son Erosión y Dilatación. Luego, sus variantes como apertura, cierre, gradiente, etc. también entran en juego.

  Este comando en sí realiza la operación de apertura que básicamente es otro nombre de erosión seguido de dilatación, es útil para eliminar ruido.

Los demás comandos que se pueden encontrar en el algoritmo de la figura 61, ya fueron mencionados en la explicación de los códigos anteriores.

```
import cv2
import numpy as np
down = False
xi, yi = 0, 0
xf, yf = 0, 0
def paint(event, x, y, flags, param):
	global down,xi,yi,xf,yf,p
	if event == cv2.EVENT_LBUTTONDOWN:
		xi, yi = x, y
		down = True
	elif event == cv2.EVENT_MOUSEMOVE and down == True:
		if down == True:
			board[:]=0
			cv2.rectangle(board,(xi,yi),(x,y),(0,255,0),3)
			xf, yf = x, y

	elif event == cv2.EVENT_LBUTTONUP:
			down = False
			p = True
def webcam(cam,color_avr):
	kernel=np.ones((5,5),np.uint8)
	def add(m,num):
		output = np.array([0,0,0],np.uint8)
		for i,e in enumerate(m):
			q = e+num
			if q>=0 and q<=255: output[i] = q
			elif q>255: output[i]=255
			else: output[i] = 0
		return output
	rangomax = add(color_avr,15)
	rangomin = add(color_avr,-15)
	print 'color_avr:\t',color_avr
	print 'rangomin :\t',rangomin
	print 'rangomax :\t',rangomax
	while(True):
		ret,frame=cam.read()
		mascara=cv2.inRange(frame,rangomin,rangomax)
		opening=cv2.morphologyEx(mascara,cv2.MORPH_OPEN,kernel)
		contours,hierarchy = cv2.findContours(opening,1,2)
		for cnt in contours:
			if np.size(cnt) > 500:
				x,y,w,h=cv2.boundingRect(cnt)
				cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),3)
				cv2.circle(frame,(x+w/2,y+h/2),5,(0,0,255),-1)
		cv2.imshow("cam",frame)
		k=cv2.waitKey(1) & 0xFF
		if k==27:
			break
	cam.release()
	cv2.destroyAllWindows()
def main():
	global board,p
	p = False
	cam = cv2.VideoCapture(0)
	board = np.zeros((int(cam.get(4)),int(cam.get(3)),3),dtype=np.uint8)
	while(True):
		ret,frame=cam.read()
		cv2.namedWindow('board')
		cv2.setMouseCallback('board', paint)
		dst = cv2.addWeighted(frame,1,board,1,0)
		cv2.imshow('board',dst)
		k=cv2.waitKey(1) & 0xFF
		if k==27 or p:
			break
	cv2.destroyAllWindows()
	if xi!=xf and yi!=yf:
		ext = frame[yi:yf,xi:xf]
		s=np.array([0,0,0])
		for i in range(np.shape(ext)[0]):
			for j in range(np.shape(ext)[1]):
```

```
                                s+=ext[i][j]
                    webcam(cam,s/((i+1)*(j+1)))

if __name__=='__main__':
        print
        print "Comandos"
        print "======================="
        print
        print "Salir: \t\t\t[ESC]\n"
         main ()
```

Figura 61. Código clasificador de Colores

(Fuente: https://robologs.net/2016/05/18/detectar-multiples-colores-con-opencv-y-python/)

De la misma forma se procedió a realizar la verificación del algoritmo a través de unos ensayos experimentales que permitían identificar su eficacia al momento de la detección en diferentes condiciones de distancia al objeto como de iluminancia.

Igualmente, como el código permitía manejar una tolerancia en el rango de RGB del color que se quería identificar, procedimos a realizar las validaciones con tres tipos de tolerancias $\pm 20, \pm 30\ y\ \pm 40$, con el fin, de determinar la diferencia de detección del algoritmo al momento de la ejecución y de esta forma analizar si es adecuado manejar tolerancias de color grandes o pequeñas de acuerdo con nuestra aplicación.

En este sentido, en la figura 62 se muestra el desarrollo de las pruebas iniciando con una tolerancia de ± 20.

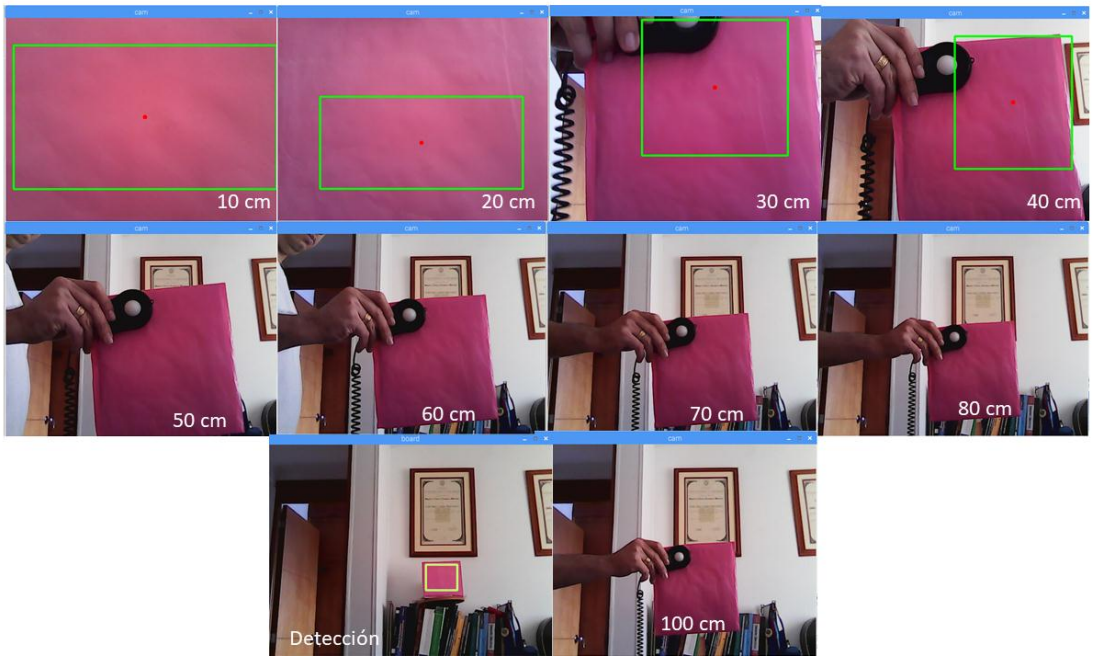

Figura 62. Ensayo de detección de colores con una tolerancia de +- 20

(Fuente: Propia)

Igualmente, en la figura 64 se presentan los resultados de este trabajo experimental.

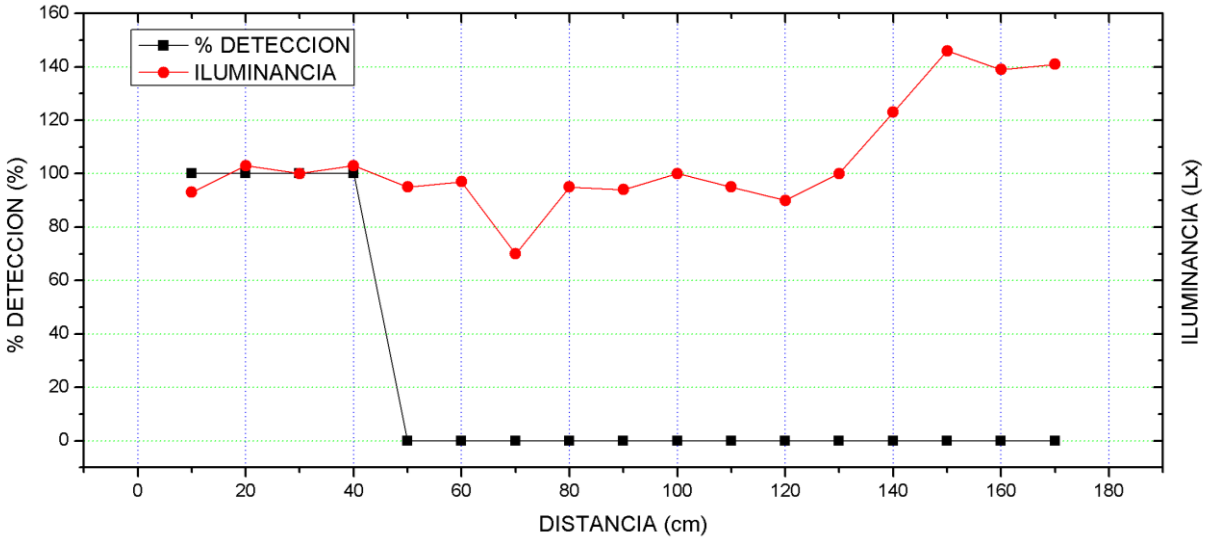

Figura 63. Comportamiento de detección del algoritmo de detección de colores con tolerancia de +- 20

(Fuente: Propia)

En la figura 64, se presenta el desarrollo experimental de validación del algoritmo de detección de colores con una tolerancia de ± 30.

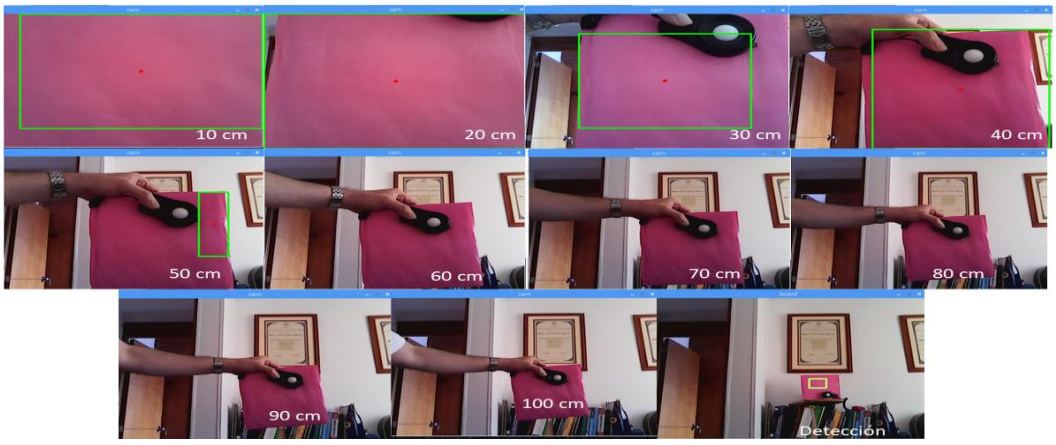

Figura 64. Figura 64. Ensayo de detección de colores con una tolerancia de +- 30

(Fuente: Propia)

Y así mismo, se presentan los resultados de este trabajo experimental con una tolerancia de ± 30.

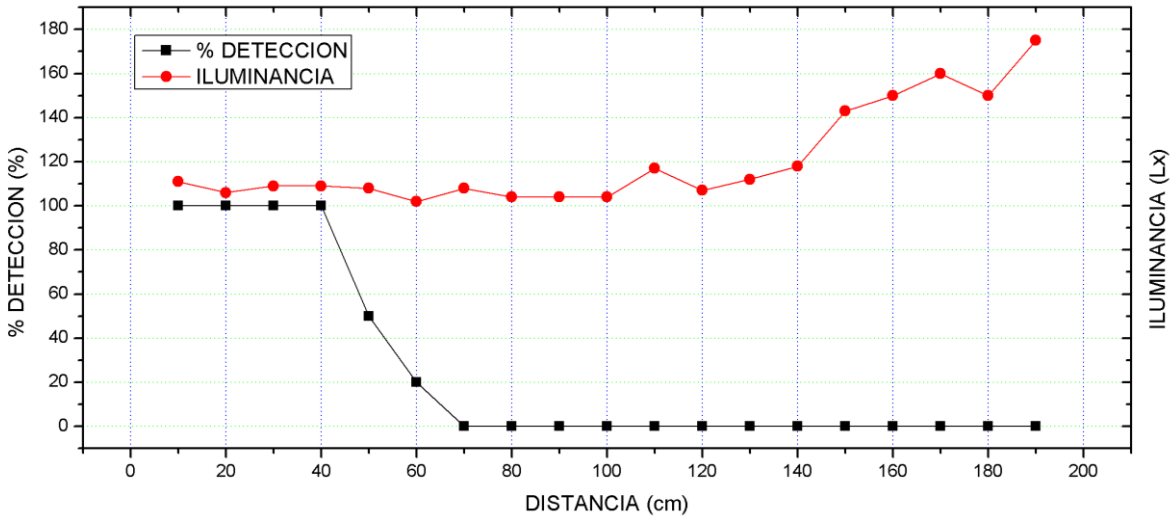

Figura 65. Comportamiento de detección del algoritmo de detección de colores con tolerancia de +- 30

(Fuente: Propia)

Y por último se hizo la validación del algoritmo con una tolerancia de ± 40 niveles de RGB, que se puede evidenciar en la figura 66.

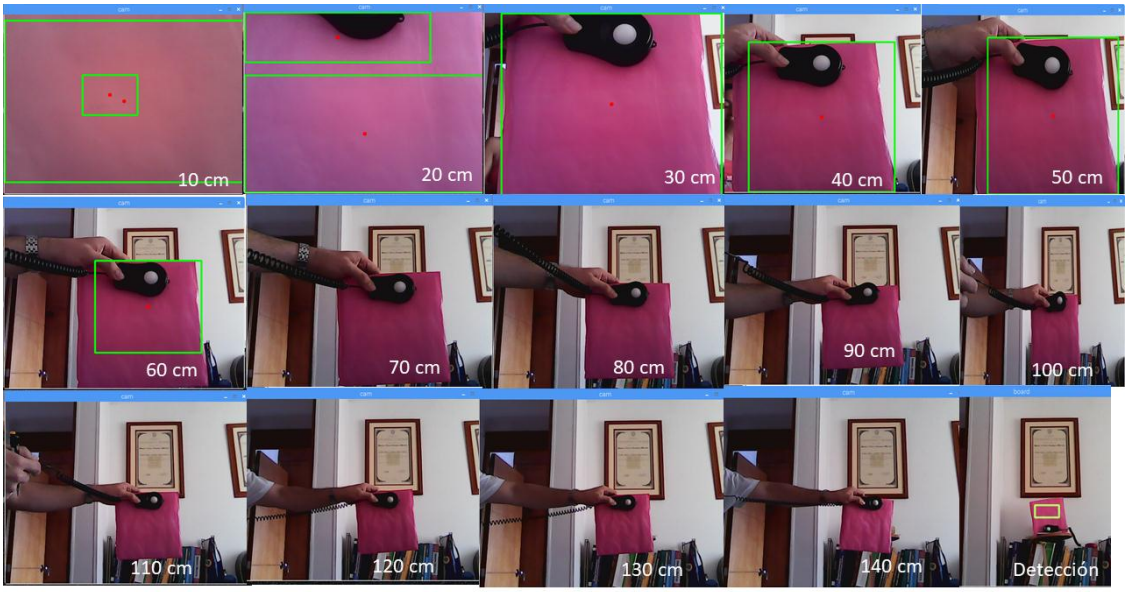

Figura 66. Ensayo de detección de colores con una tolerancia de +- 40 niveles de RGB

(Fuente: Propia

Y los resultados que arrojo de este ensayo experimental son los que se visualizan en la figura 67.

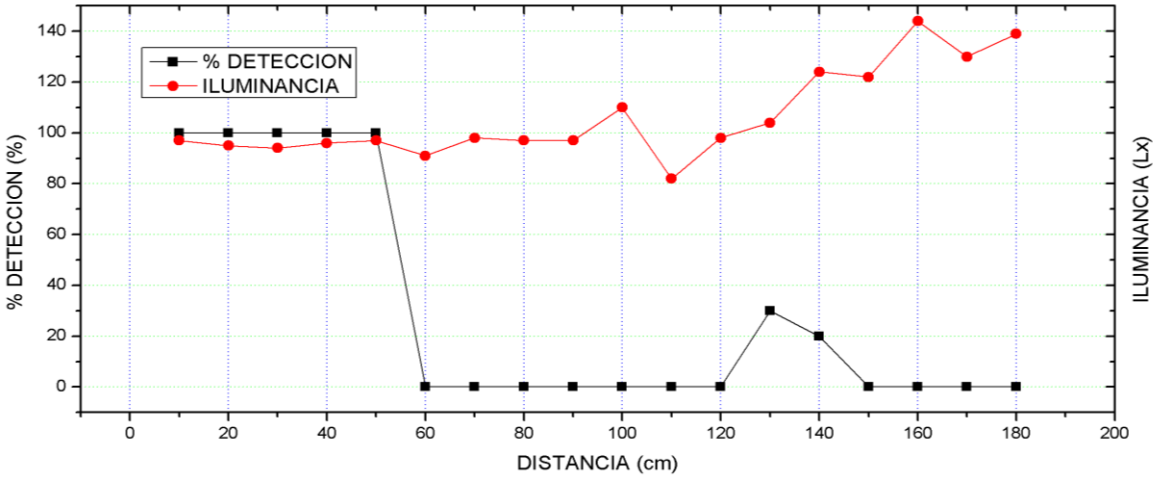

Figura 67. Comportamiento de detección del algoritmo de detección de colores con tolerancia de +- 40

(Fuente: Propia)

Posteriormente, se procedió a graficar en un solo campo los resultados de las validaciones de este algoritmo con las diferentes tolerancias trabajadas, en la figura 68, se representa el comportamiento del algoritmo con las tres tolerancias de niveles de RGB mencionadas.

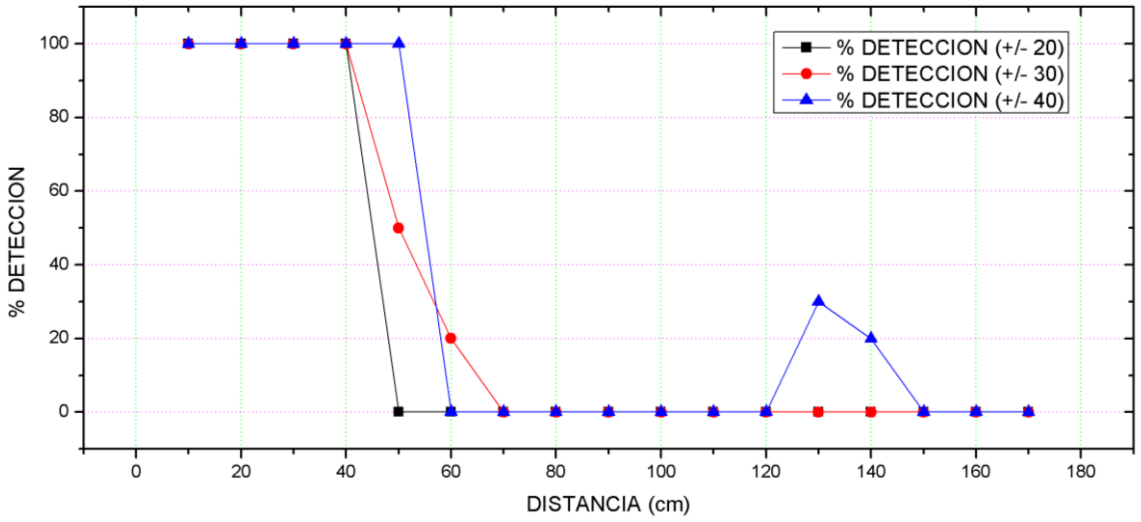

Figura 68. Consolidación de los resultados de la detección de colores con las tres tolerancias estudiadas.

(Fuente: Propia)

3.3.4 Detección de Círculos

Teniendo en cuenta que la morfología de la mancha de la "alternaría" escogida para este estudio según literatura, se presenta en figuras asemejadas a formas circulares y a aureolas amarillas concéntricas, se procedió a buscar un código o algoritmo que identificara este tipo de formas geométricas, en la literatura existen varios códigos de este tipo, para este trabajo se analizó el que se encuentra en la figura 69 [69].

```
import numpy as np
import argparse
import cv2
import time

cap = cv2.VideoCapture(0)

while(True):

        ret, frame = cap.read()
        output = frame.copy()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray,(5,5),0);
        gray = cv2.medianBlur(gray,5)
        circles=cv2.HoughCircles(gray, cv2.cv.CV_HOUGH_GRADIENT, 1, 200, param1=55,
param2=35, minRadius=0, maxRadius=60)

        if circles is not None :

                circles = np.round(circles[0, :]).astype("int")

                for (x, y, r) in circles:

                        cv2.circle(output, (x, y), r, (0, 255, 0), 4)
                        cv2.rectangle(output, (x - 5, y - 5), (x + 5, y + 5), (0,
128, 255), -1)
                        cv2.circle(gray, (x, y), r, (0, 255, 0), 4)
                        cv2.rectangle(gray, (x - 5, y - 5), (x + 5, y + 5), (0,
128, 255), -1)

                        print "Column Number: "
                               print x
                        print "Row Number: "
                        print y
                        print "Radius is: "
                        print r

                        cv2.imshow('gray',gray)
                        cv2.imshow('frame',output)

        if cv2.waitKey(1) & 0xFF == ord('q'):
           break


cap.release()
cv2.destroyAllWindows()
```

Figura 69. Código de detección de figuras circulares en la imagen

(Fuente: [67])

Para su despliegue el código cuenta con una serie de comandos extraídos de las librerías de OpenCV, y que realizan unas funciones específicas como son:

- gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY: Este comando permite convertir una imagen a escala de niveles de grises
- cv2.GaussianBlur: Este comando permite aplicar un filtro de desenfoque tipo gaussiano

- cv2.medianBlur(gray,5): Este comando aplica un filtro tipo mediano que suaviza la imagen usando con Ksize x Ksize de apertura.
- cv2.HoughCircles: Es el comando principal de este código, permite identificar de una imagen de estudio los contornos de patrones que más se asemejan a una figura circular, este comando usa el método de Hough Gradient, que utiliza información de degrado de los bordes.
- np.round: Este comando permite redondear una matriz de valores decimales
- cv2.circle(output, (x, y), r, (0, 255, 0), 4: Este comando aplica una función de dibujo y permite dibujar un circulo de un color y espesor característico, dentro de la imagen de estudio.
- cv2.rectangle: Este comando aplica una función de dibujo y permite dibujar un rectángulo de un color y espesor característico, dentro de la imagen de estudio.

Este código permite ajustar el radio de detección de las figuras circulares que se desean detectar, para este caso en particular se realizó la verificación tomando un radio máximo de detección de las figuras circulares de 60 px, con el fin, de evitar ruidos de detección de figuras que estaban por fuera del rango del tamaño de la mancha del hongo en cuestión.

Igualmente, como se ha hecho con los otros códigos que se han estudiado se procedió a realizar la evaluación del funcionamiento de este, realizando una prueba experimental en donde determinaba el porcentaje de detección en relación con la distancia e iluminancia que se tenía para el procesamiento de las imágenes.

En la figura 70, se puede evidenciar las etapas de la validación del algoritmo.

Figura 70. Validación de algoritmo de detección de figuras geométricas circulares

(Fuente: Propia)

Posteriormente, los resultados de este código de detección de figuras circulares se graficaron en la figura 71.

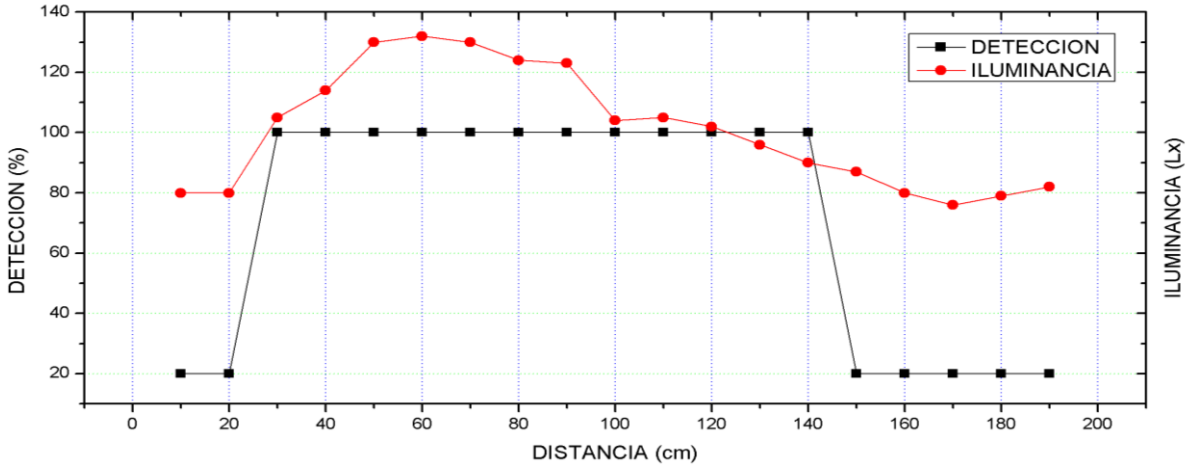

Figura 71. Figura 71. Resultados de detección del código de detección de figuras circulares

(Fuente Propia)

3.3.5 Comparación de los algoritmos analizados encontrados en la literatura

Lo que se pretende con esta recopilación de algoritmos que se han encontrado en diferentes fuentes de literatura, es comparar sus funcionamientos y poder determinar los algoritmos más eficaces para la aplicación específica que se tiene en el campo de la agricultura. De acuerdo con esto, se procedió a recopilar o consolidar en una sola gráfica los comportamientos de detección de cada uno de los códigos trabajados como se puede ver en la figura 72, es de anotar que algunos códigos tuvieron que ser modificados desde los encontrados en las fuentes originales ya que por cuestiones de sinapsis, versiones y actualizaciones de los software y hardware no corrían. No obstante, todos los algoritmos se pusieron en marcha y se comprobó el funcionamiento de cada uno de primera mano. Esta actividad consumió gran cantidad de tiempo, pero era importante comprobar personalmente que estos algoritmos funcionaban y que si cumplían la aplicación con la que fueron desarrollados.

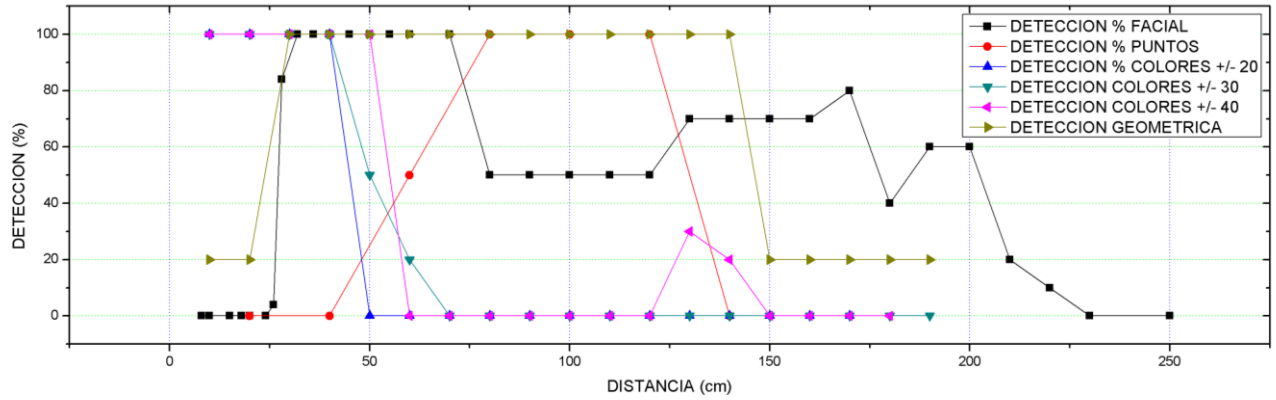

Figura 72. Consolidad de resultados de detección de los diferentes códigos estudiados

Para poder encontrar el algoritmo adecuado para implementar en la aplicación de agricultura de precisión a partir de reconocimiento de patrones de visión artificial, se deben tener en cuenta dos factores principalmente, primero, que el algoritmo no tenga un gran coste computacional, ya que se quiere implementar directamente en el open hardware (Raspberry Pi 2) y tener un video en tiempo real de la detección del hongo. Segundo, que la distancia de detección sea prudencial para que se pueda detectar a través del vuelo de reconocimiento de un drone.

En este orden de ideas y de acuerdo con la figura 72, se escogieron dos tipos de algoritmos: el primero fue la detección geométrica de círculos, debido a que la distancia de detección fue desde 40cm hasta 140 cm. El segundo código que se escogió fue el de detección de colores con una tolerancia en los niveles de RGB del ± 20 ya que la diferencia con las demás tolerancias no fue

significativa; es más, con una tolerancia de ± 40 ya se comienza a presentar ruidos a los 130 cm de distancia que no son favorables para el proceso de detección.

Igualmente, se compara el resultado de la combinación de estos dos algoritmos, con un algoritmo desarrollado a través de recursos de inteligencia artificial, usando seleccionadores en cascada con base Haar, que según literatura ha tenido muy buenas prestaciones en la detección de objetos relacionados con el rostro humano.

Este trabajo busca generar o desarrollar un nuevo código partiendo de los encontrados en las fuentes y realizar su evaluación tanto en el ambiente controlado como en el campo real.

3.3.6 Desarrollo del algoritmo de detección de Manchas de "Alternaria" en tiempo real

Como se mencionó anteriormente, a continuación se procede a mostrar y explicar los procedimientos y códigos desarrollados con base a las cualidades identificadas en los códigos de referencia, y que nos permiten pronosticar una correcta aplicación en la detección de manchas en las hojas del lulo producidas por algunos tipos de hongos como la "alternaría", este código debe tener dos cualidades principales: que tenga un coste computacional muy bajo con el que se pueda aplicar directamente al open hardware y que tenga una muy buena distancia de detección y robustez al momento de la identificación de los patrones de la mancha del hongo.

Como se vio en los análisis y figuras que se relacionaron anteriormente, los dos códigos que tienen las mejores características prestacionales para esta aplicación son: los que son generados a partir de la segmentación de características geométricas y los que aplican alguna técnica de Inteligencia Artificial adicional, que para este trabajo son los algoritmos con Haar Cascade. En ese orden de ideas, a continuación, se presentarán los dos procedimientos y algoritmos que se desarrollaron en pro de identificar cuál de los dos es el más conveniente para este trabajo, no obstante, todas las referencias, análisis y comparaciones, desarrollados en el presente estudio, son importantes, como memoria de la implementación de este tipo de técnicas en casos de identificación de enfermedades en cultivos como el lulo.

3.3.6.1 Procedimiento a partir de la segmentación e identificación de figura geométricas

Como se ilustró anteriormente, respecto a la identificación y caracterización de algunas de las enfermedades presentes en el lulo, unas de las particularidades más relevantes de la enfermedad (Mancha de Alternaria) que se tomó como piloto para realizar este trabajo, es una mancha en las hojas de forma geométrica semicircular y de color característico, evidenciada en la mayoría de los casos estudiados. Estas características sirvieron como base para seleccionar y adecuar un algoritmo, que segmentará e identificará un cierto tipo de figura geométrica semicircular y un cierto color determinado.

En la figura 73, se presenta el código desarrollado, el cual, dentro de su funcionalidad permite identificar tanto un color específico como una morfología geométrica de mancha semicircular, en donde, si el código identifica estas dos señales en una misma ubicación, se puede decir, que hay presencia de la patología.

```
import cv2
import numpy as np

down = False
xi, yi = 0, 0
xf, yf = 0, 0

def paint(event, x, y, flags, param):
    global down,xi,yi,xf,yf,p
    if event == cv2.EVENT_LBUTTONDOWN:
        xi, yi = x, y
        down = True
    elif event == cv2.EVENT_MOUSEMOVE and down == True:
        if down == True:
            board[:]=0
            cv2.rectangle(board,(xi,yi),(x,y),(0,255,0),3)
            xf, yf = x, y

    elif event == cv2.EVENT_LBUTTONUP:
            down = False
            p = True
def webcam(cam,color_avr):
    kernel=np.ones((5,5),np.uint8)
    def add(m,num):
        output = np.array([0,0,0],np.uint8)
        for i,e in enumerate(m):
            q = e+num
            if q>=0 and q<=255: output[i] = q
            elif q>255: output[i]=255
            else: output[i] = 0
        return output
```

```
    rangomax = add(color_avr,20)
    rangomin = add(color_avr,-20)
    print 'color_avr:\t',color_avr
    print 'rangomin :\t',rangomin
    print 'rangomax :\t',rangomax
    while(True):
         ret,frame=cam.read()
        mascara=cv2.inRange(frame,rangomin,rangomax)
        opening=cv2.morphologyEx(mascara,cv2.MORPH_OPEN,kernel)
        contours,hierarchy = cv2.findContours(opening,1,2)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray,(5,5),0);
        gray = cv2.medianBlur(gray,5)
        circles=cv2.HoughCircles(gray, cv2.cv.CV_HOUGH_GRADIENT, 1, 300, param1=55, param2=35, minRadius=0,
maxRadius=200)
        if circles is not None:
                circles = np.round(circles[0, :]).astype("int")
                for (x, y, r) in circles:
                    cv2.circle(frame, (x, y), r, (0, 255, 0), 4)
                    cv2.rectangle(frame, (x - 5, y - 5), (x + 5, y + 5), (0, 128, 255), -1)
                    cv2.circle(gray, (x, y), r, (0, 255, 0), 4)
                    cv2.rectangle(gray, (x - 5, y - 5), (x + 5, y + 5), (0, 128, 255), -1)
                    print "Column Number: "
                    print x
                    print "Row Number: "
                    print y
                    print "Radius is: "
                    print r
        for cnt in contours :
            if np.size(cnt) > 500:
                x,y,w,h=cv2.boundingRect(cnt)
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),3)
                cv2.circle(frame,(x+w/2,y+h/2),5,(0,0,255),-1)
        cv2.imshow("gris",gray)
        cv2.imshow("cam",frame)
        k=cv2.waitKey(1) & 0xFF
        if k==27:
            break
    cam.release()
    cv2.destroyAllWindows()
def main():
    global board,p
    p = False
    cam = cv2.VideoCapture(0)
    board = np.zeros((int(cam.get(4)),int(cam.get(3)),3),dtype=np.uint8)
    while(True):
        ret,frame=cam.read()
        cv2.namedWindow('board')
        cv2.setMouseCallback('board', paint)
        dst = cv2.addWeighted(frame,1,board,1,0)
        cv2.imshow('board',dst)
        k=cv2.waitKey(1) & 0xFF
        if k==27 or p:
            break
    cv2.destroyAllWindows()
    if xi!=xf and yi!=yf:
        ext = frame[yi:yf,xi:xf]
        s=np.array([0,0,0])
        for i in range(np.shape(ext)[0]):
            for j in range(np.shape(ext)[1]):
                s+=ext[i][j]
        webcam(cam,s/((i+1)*(j+1)))
```

Figura 73. Código de combinación de detección de figuras circulares y colores específicos

(Fuente: [67])

En la sección de pruebas y verificaciones, se mostrará el trabajo experimental que se desarrolló para evaluar este código en un ambiente controlado, donde se procedió a variar la distancia que hay entre el elemento captador de la imagen y el objeto a detectar, tomando datos de la iluminancia que se presenta al momento del estudio. Para determinar el porcentaje de detección del algoritmo, se tomó el tiempo de detección en un intervalo de 2 minutos, este procedimiento de validación de software de visión artificial es similar a los usados en otros trabajos como [66], [70], [67].

3.3.7 Procedimiento con clasificadores en Cascada basados en características Haar y HOG

La detección de objetos usando clasificadores en cascada basados en características de Haar es un método de detección de objetos propuesto por Paul Viola y Michael Jones [66], [67], [70], consiste en un aprendizaje automático donde la función en cascada es entrenada a partir de muchas imágenes positivas y negativas que posteriormente son usadas también para su verificación. Como se dijo anteriormente el algoritmo necesita muchas imágenes positivas y negativas para entrenar el clasificador para esto se deben extraer características especiales de las imágenes, en tal sentido, se utilizan funciones Haar que son el filtro convolucional.
Estos filtros pueden ser calculados eficientemente sobre la imagen integral, son selectivos en la orientación espacial y de frecuencia, y pueden ser modificados en escala y orientación, en la figura 74 se muestran algunos filtros usados para la extracción de características de las imágenes [70].

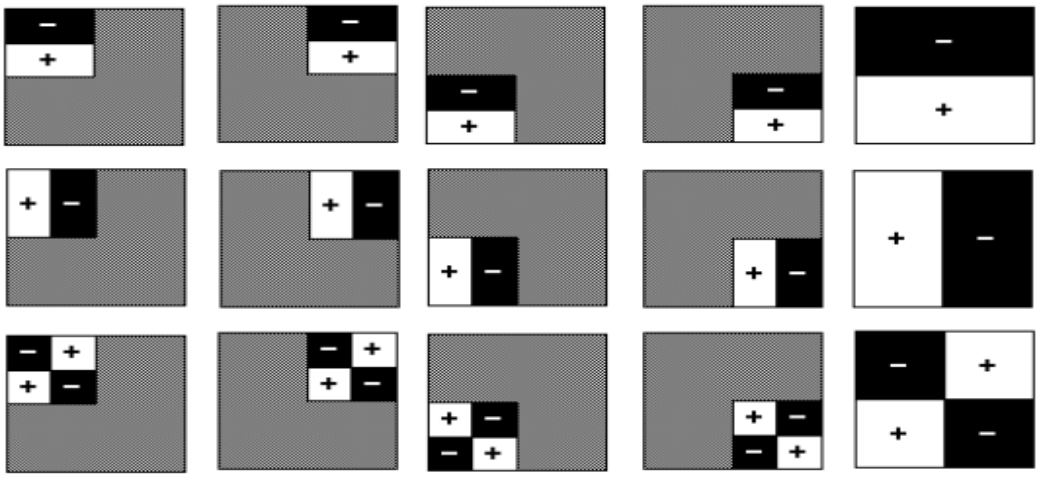

Figura 74. Kernel Haar para extracción de características de las imágenes

(Fuente: [68])

Estas funciones de filtros o kernel con bases Haar, realizan una codificación diferencia de intensidades de la imagen, provocando características de contorno, puntos y líneas, mediante la captura de contraste entre regiones.

Así mismo, se pueden usar también filtros de Histogramas de Gradientes Orientados (HOG), que en algún momento pueden reemplazar los filtros Haar, las dos técnicas son muy usadas en el reconocimiento de patrones.

Para poder implementar este descriptor HOG, la imagen es segmentada en un conjunto de celdas uniformes. El algoritmo estima la orientación del gradiente de los pixeles que conforman una celda y reúne la información en un histograma de orientaciones de N clase. Este conjunto de histogramas es utilizado para describir el objeto a seguir.

En la figura 75, se presentan las tres etapas de los descriptores HOG

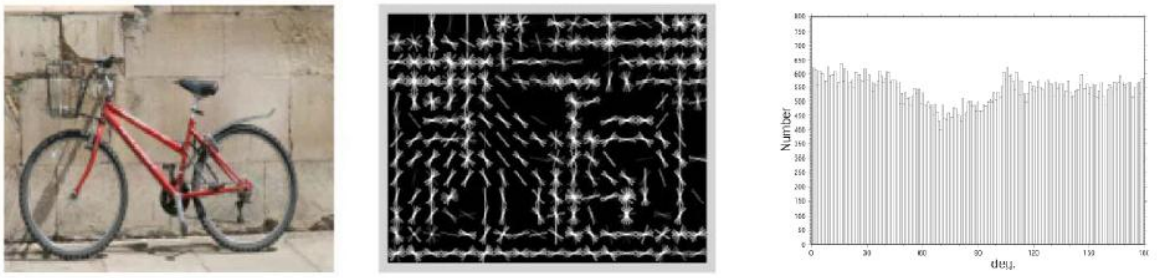

Figura 75. Descriptores HOG sobre una imagen.

a) Imagen original b) Imagen que muestra la orientación del gradiente c) Histograma de orientación del Gradiente

(Fuente: [68])

Para realizar el cálculo del gradiente se filtra la imagen usando dos máscaras, una para calcular la componente G_x y la otra para calcular la componente G_y del gradiente.

Las máscaras usadas son $maskH = [1\ 0 - 1]^T$ y $maskV = [1\ 0 - 1]$ respectivamente. Para determinar la magnitud y el ángulo del gradiente se utilizan las siguientes ecuaciones [71].

$$|G(m,n)| = \sqrt{G_x^2(m,n) + G_y^2(m.n)}$$

$$< G(m,n) = arc\left(\frac{G_y(m,n)}{G_x(m,n)}\right)$$

En la etapa de clasificación dentro del algoritmo de detección se asigna un conjunto de características dado a una clase con la que se encuentra una mayor similitud, de acuerdo con un modelo inducido durante la fase de entrenamiento.

Para esta fase del algoritmo de detección se implementó el proceso de Boosting, el cual, fue introducido por [72], y es un método de clasificación que combina varios clasificadores básicos para formar un único clasificador más complejo, preciso y eficaz. Este algoritmo precisa que varios clasificadores sencillos, cada uno de ellos con una precisión superior, pueden combinarse para conseguir un clasificador de mayor precisión, siempre y cuando se tenga un número suficiente de muestras y fases de entrenamiento. La aplicación de estos clasificadores en cascada ha permitido obtener buenos resultados en la detección de rostros reportados en varios estudios.

En la figura 76, se presenta un esquema de un clasificador en cascada [66], [67], [70].

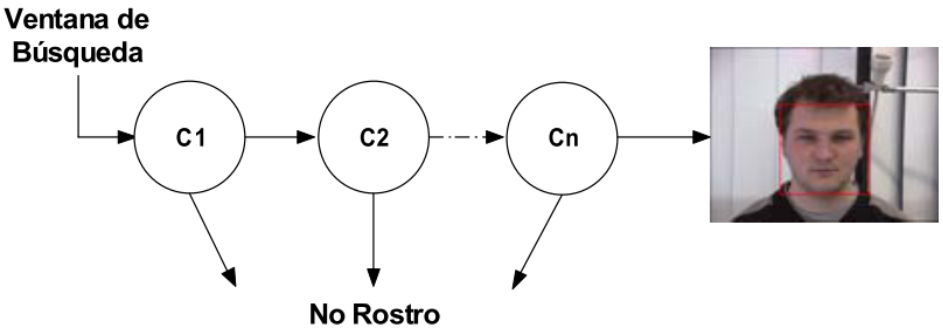

Figura 76. Esquema de un clasificador en cascada

(Fuente: [67])

Para aplicar la técnica de boosting, primero se debe establecer un algoritmo base de aprendizaje sencillo, que será ejecutado varias veces, para generar diversos clasificadores base. Para el entrenamiento del clasificador base, se emplea en cada iteración un subconjunto de diferentes muestras y pesos. Finalmente, estos clasificadores base se combinan en un solo clasificador que deba ser más preciso y robusto.

En relación con los clasificadores base que se utilicen, las distribuciones que se emplean para entrenarlos y el modo de combinarlos, podrán darse diversas clases de algoritmos genéricos de boosting.

A continuación se ilustra el algoritmo usado en la metodología de boosting, el cual, se encuentra planteado en [72].

Dado un conjunto de imágenes $(x_1, y_1), \dots, (x_n, y_n)$ donde $y_i = 0, 1$ para muestras negativas y para muestras positivas respectivamente.

Inicializar los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0, 1$, donde m es el número de muestras negativas y l es el número de muestras positivas.

Para t=1, …, T:

1. Normalizar los pesos $w_{t,i} \longleftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
2. Seleccionar el mejor clasificador base respecto al peso del error:

$$\in = \min \sum_i w_i |h(x_i, f, p, \theta) - y_i|$$

3. Definir $h_t(x) = h(x, f_t, p_t, \theta_t)\ donde\ f_t, p_t\ y\ \theta_t$ son usadas para minimizar $\in_t$.
4. Actualizar los pesos: $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ donde $e_i = 0$ si la muestra x_i es clasificada correctamente, o $e_i = 1$ en otro caso, $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
5. El clasificador robusto final es:

$$C(x) = \begin{cases} 1 \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0\ en\ otro\ caso \end{cases}$$

Para la implementación tanto del proceso de segmentación a través de los kernel como del entrenamiento y puesta en marcha del algoritmo en la Raspberry Pi, se usaron las librerías que tiene para este fin OpenCV. OpenCV cuenta con un entrenador y un detector, no obstante, cabe la posibilidad de desarrollar un clasificador propio, con base a los requerimientos particulares que se tengan identificados.

Es decir, desarrollar seleccionadores para identificar objetos distintos a las caras o rostros, por ejemplo, autos, aviones, etc.

Para este trabajo, se procedió a desarrollar un seleccionador o clasificador particular, ya que el objeto de este, debe ser la identificación de las manchas en las hojas del cultivo del lulo. A

continuación, se ilustra el procedimiento que se realizó para entrenar y desarrollar este seleccionador.

Como primer paso se debe tener las imágenes tanto positivas como negativas para el respectivo entrenamiento, las imágenes positivas son aquellas en donde se encuentra el objeto a detectar de distintas formas y vistas, y las imágenes negativas son aquellas que no tienen ninguna relación con el objeto que se va a identificar, es decir, es una cantidad de imágenes en donde no se encuentra el objeto a identificar o seleccionar. Para realizar una buena fase de entrenamiento se requiere el mayor número de imágenes que sea posible, entre más imágenes se obtengan para el entrenamiento, más eficiente, eficaz y robusto será el programa de detección.

En la figura 77, se muestra un resumen de las imágenes positivas que se tomaron para hacer el entrenamiento del algoritmo, para aumentar la cantidad de estas se optó por duplicar a algunas y de esta forma aumentar hasta 200 las imágenes positivas.

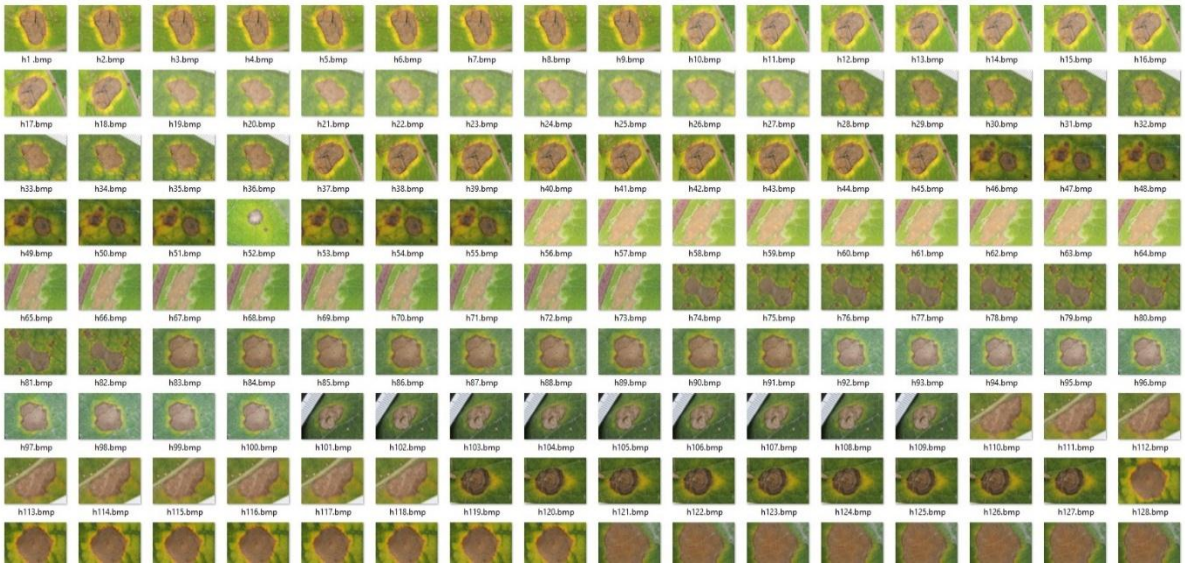

Figura 77. Muestras de imágenes positivas para el entrenamiento

(Fuente: Propia)

Es importante resaltar, que cada una de estas imágenes se tuvieron que editar, primero cortando el área de interés donde se ubica el objeto a detectar y segundo redimensionando la imagen, con el fin, de estandarizar el tamaño de estas a 640 x 480 px.

De esta misma forma, se usó un repositorio de imágenes negativas usado en la literatura. Una muestra de ellas se puede ver en la figura 78.

Figura 78. Muestras de imágenes negativas para el entrenamiento

(Fuente: https://www.cs.auckland.ac.nz/~m.rezaei/Downloads.html)

Con esta información se procedió a desarrollar el archivo del clasificador combinado, usando la metodología divulgada por [73], que indica que después de tener las imágenes organizadas en carpetas separadas debidamente marcadas, se ejecuta una serie de archivos por lotes, el cual, inicialmente permite obtener un archivo de texto, con características de cada imagen negativa (Figura 79), posteriormente, se corta y marcan las imágenes positivas. Para realizar esta actividad se usa la aplicación de Objectmarker, esto permite obtener un archivo de texto con los nombres de las imágenes positivas y la ubicación de los objetos a detectar en cada imagen (Figura 80), a continuación, con la información obtenida se halla un archivo vectorial (.vec), ejecutando un archivo de lotes denominado samples_creation.bat, después se ejecuta el archivo de lotes haar training.bat, el cual, toma el vector generado por la información de las imágenes positivas y el archivo de texto de las imágenes negativas, para ejecutar el número de etapas de entrenamiento que se le ha asignado y de esta forma arrojar cada uno de los algoritmos Adaboost producidos en la etapa de entrenamiento [67], [70], Por último, se debe empaquetar en un archivo XML todos los algoritmos producidos en cada una de las etapas del entrenamiento para esto se usa el archivo de lotes convert.bat, que se muestra en la figura 81, y que posteriormente será implementado en Python para el despliegue y ejecución de la detección.

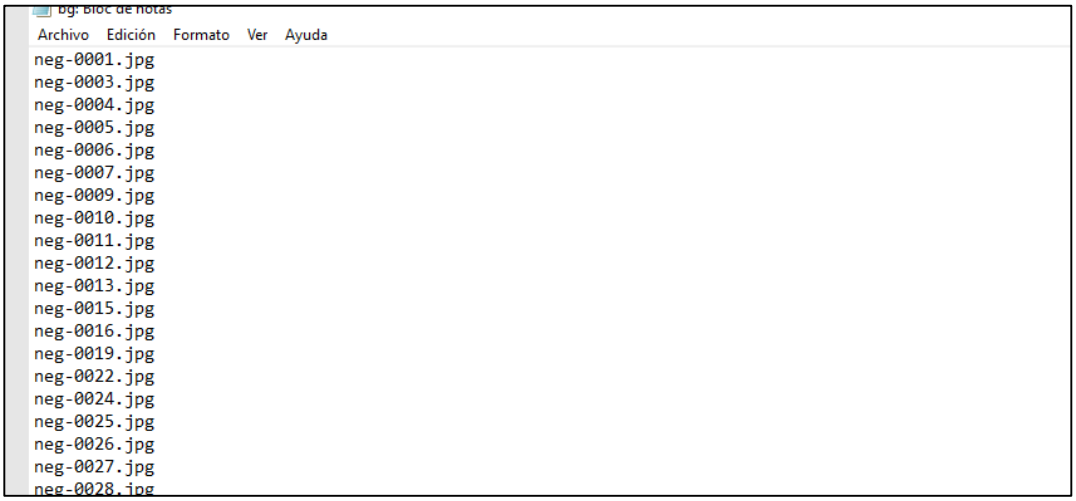

Figura 79. Archivo de Texto con características de las imágenes negativas

(Fuente: Propia)

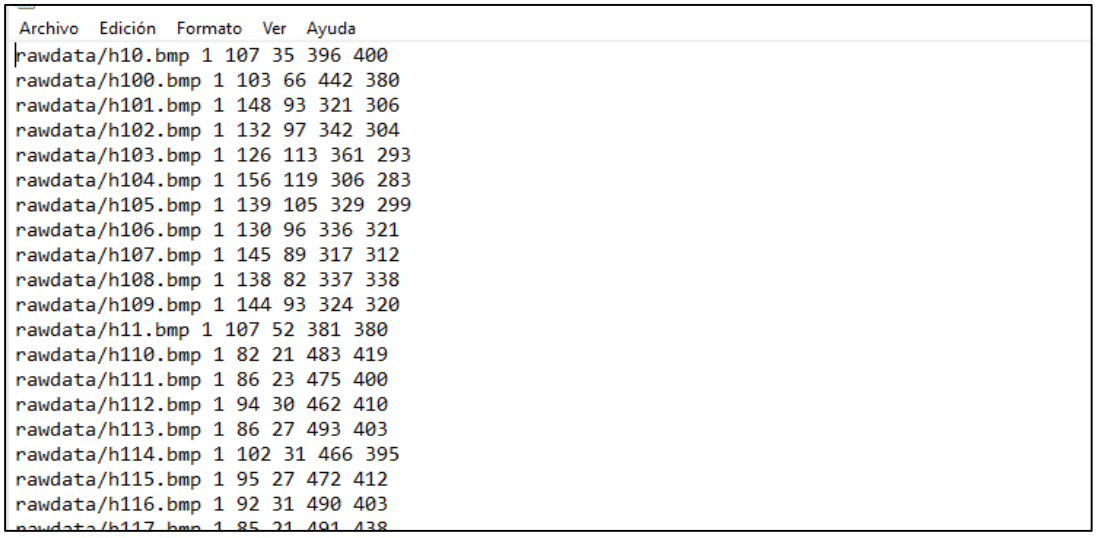

Figura 80. Archivo de texto con el nombre de las imágenes positivas y ubicación del objeto

(Fuente: Propia)

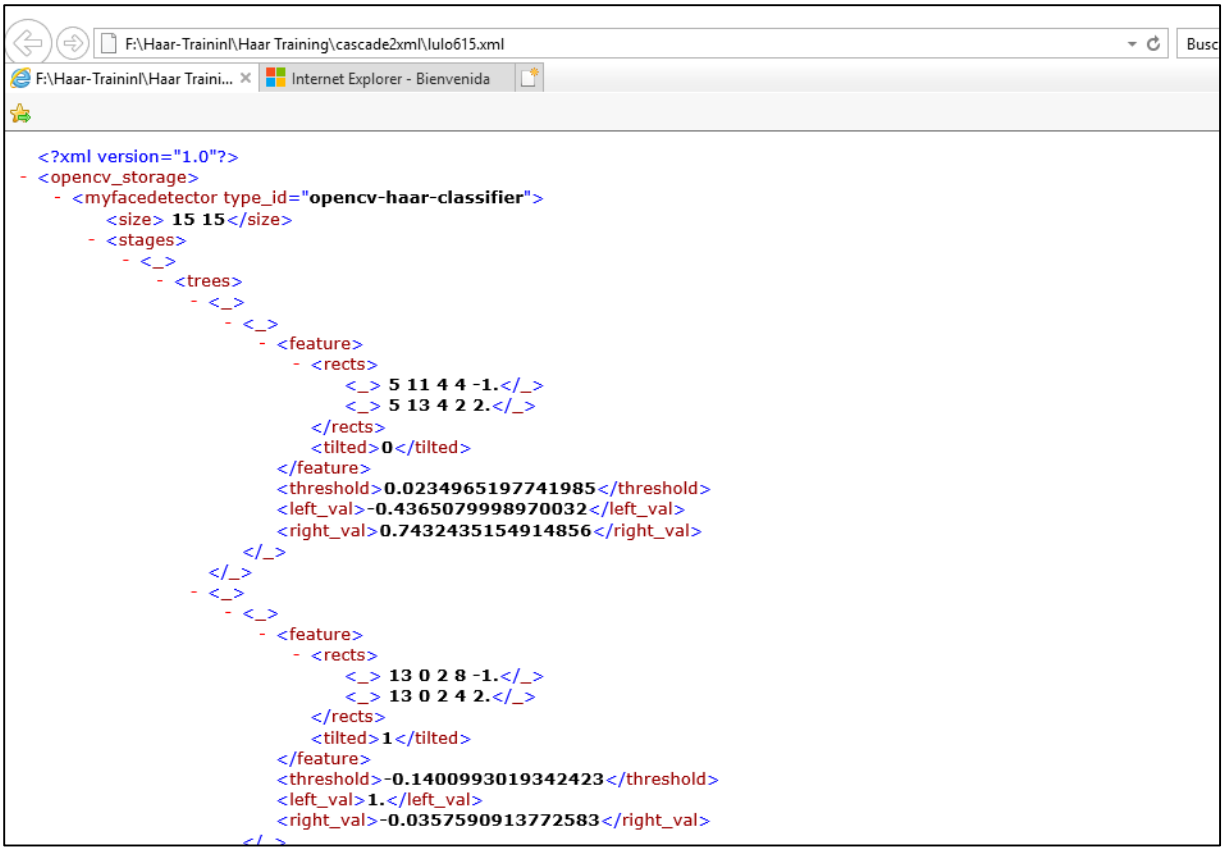

Figura 81. Archivo xml del detector encontrado a través de la segmentación de las imágenes positivas y negativas disponibles

(Fuente: Propia)

Con el archivo XML obtenido, se procede a implementar el código fuente del programa en Python, como se ve en la figura 82.

reconolulo.py - /home/pi/luloprueba/reconolulo.py (2.7.13)

File Edit Format Run Options Window Help

```
import cv2
import numpy

face_cascade = cv2.CascadeClassifier('lulo615.xml')
cap = cv2.VideoCapture(0)

while(True):
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)

    cv2.imshow('Haar',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Figura 82. Implementación del archivo xml en el programa desarrollado en Python

(Fuente: Propia)

Igualmente, hay diversas formas en la literatura para obtener el archivo .XML, el cual, tiene el algoritmo de aprendizaje consolidado.

A continuación, mostraremos el procedimiento para obtener este archivo usando algunas rutinas en Matlab®, asimismo, mostraremos la evaluación de estos algoritmos usando indicadores o elementos que permite de tener un dato sobre la eficiencia y eficacia del algoritmo al momento de identificar patrones.

El procedimiento de etiquetado, ya que es un proceso de aprendizaje supervisado, y de entrenamiento inicia de la siguiente manera:

Primero se procede a abrir la APP de Matlab® training image labeler, el cual, es una APP que reemplaza Objectmarker del procedimiento anterior, en esta etapa igual que en el anterior caso se procede a abrir cada una de las imágenes de entrenamiento del algoritmo y se inicia el etiquetado del elemento que se quiera identificar. Al finalizar esta actividad se debe guardar la sesión y exportar un archivo plano .mat en donde se encontrará la región en donde se ubicaron los elementos que se quieren identificar en cada imagen positiva, posteriormente, este mismo procedimiento se realiza para las imágenes que se van a utilizar para validar el algoritmo.

En la figura 83, se presenta una imagen sobre cómo se realiza el proceso de etiquetado de las imágenes positivas y de validación.

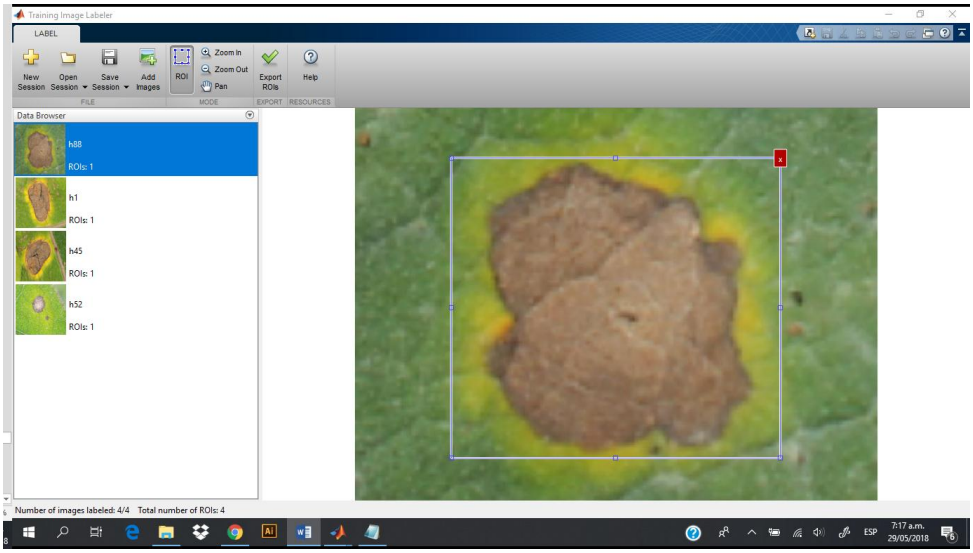

Figura 84. Script de entrenamiento en Matlab usando el algoritmo de Adaboost

Teniendo ya etiquetadas las imágenes positivas de entrenamiento y las de validación y seleccionadas y separadas en carpetas, se procede a ejecutar las siguientes líneas en Matlab, que se pueden apreciar en la figura 84.

```
load('positivas.mat');
positiveFolder = 'positive';
negativeFolder = 'negative';
detectorName = 'lulomatlab.xml';
trainCascadeObjectDetector(detectorName,Gtrain,negativeFolder,...
      'ObjectTrainingSize', 'Auto', 'NegativeSamplesFactor', 2,...
      'NumCascadeStages', 7, 'FalseAlarmRate', 0.15,...
      'TruePositiveRate', 0.995,'FeatureType', 'HOG');
```

Figura 85. Figura 85. Script de entrenamiento en Matlab usando el algoritmo de Adaboost

(Fuente: Gómez 2015)

En específico. Como se observa, el algoritmo de entrenamiento en cascada tendrá el nombre de `lulomatlab.xml`, el cual, es la base del algoritmo de aprendizaje Adaboost y se puede adicionar al código que se vio anteriormente que está desarrollado en Python, que puede ser implementado en el open hardware de Raspberry Pi.

Adicionalmente, se usan de dos códigos: uno que permite realizar pruebas con las imágenes archivadas el proceso de entrenamiento y el otro permite validar y obtener datos de la eficacia y eficiencia del algoritmo desarrollado.

En la figura 85, se pueden apreciar los códigos de prueba del algoritmo en cascada obtenido y visualizar el etiquetado del objeto detectado por el algoritmo, como se puede apreciar en la figura 86.

```
detector = vision.CascadeObjectDetector(ArchivoDetector);
D = struct('imageFilename',{},'objectBoundingBoxes',{});
for im= 1:length(G)
    img = imread(G(im).imageFilename);
    D(im).imageFilename = G(im).imageFilename;
    D(im).objectBoundingBoxes = step(detector, img);

    % Visualización
    if visualizacion
        detectedImg = insertObjectAnnotation(img, 'rectangle',...
            D(im).objectBoundingBoxes, 'Deteccion','Color','red');
        detectedImg = insertObjectAnnotation(detectedImg, 'rectangle',...
            G(im).objectBoundingBoxes, 'Entrenamiento','Color','blue');
        imshow(detectedImg);
        pause
    end
end
%Prueba
detector = vision.CascadeObjectDetector(detectorName);
img = imread('positive/h195.bmp');
bbox = step(detector, img);
detectedImg = insertObjectAnnotation(img, 'rectangle', bbox, 'Mancha de
Alternaria');
figure:imshow(detectedImg);
```

Figura 86. Código de prueba del algoritmo Adabost desarrollado en Matlab

(Fuente: Gómez 2015)

Figura 87. Imagen con etiqueta de detección generada por el código Adabost en Matlab

(Fuente: Propia)

El código usado para la validación del algoritmo Adaboost se presenta en la figura 87, en el cual se calcula tres parámetros de interés: medida F1 del Detector, la precisión del detector y la sensibilidad o recall del detector.

```
function [F1, P, R] = evaluandoDetector(G,D)

%   F1 = Medida F1 del detector
%   P = Precisión del detector
%   R = Sensibilidad (Recall) del detector

sumMejorTraslapeD = 0;
Ndet = 0;
sumMejorTraslapeG = 0;
Nobj = 0;
for im=1:length(D)
    % Calculando matriz de traslape
    IndiceTraslape =
bboxOverlapRatio(G(im).objectBoundingBoxes,D(im).objectBoundingBoxes);

    ndet = size(D(im).objectBoundingBoxes,1);
    Ndet = Ndet + ndet;
    nobj = size(G(im).objectBoundingBoxes,1);
    Nobj = Nobj + nobj;
    if ndet>0 && nobj>0
        % Calculando Precisión
        sumMejorTraslapeD = sumMejorTraslapeD + sum(max(IndiceTraslape,[],1));
        % Calculando Sensibilidad
        sumMejorTraslapeG = sumMejorTraslapeG + sum(max(IndiceTraslape,[],2));
    end
end
P = sumMejorTraslapeD/Ndet;
R = sumMejorTraslapeG/Nobj;
F1 = 2*P*R/(P+R);

% Validación con otras imagenes
load('positivastest.mat');
Dtest = detectandoSobreG(detectorName,positivasvalidacion2, true);
[F1_test, P, R] = evaluandoDetector(positivasvalidacion2,Dtest)
```

Figura 88. Código de validación desarrollado en Matlab del algoritmo de entrenamiento Adaboost

(Fuente: Gómez, 2015)

A continuación, ampliaremos un poco la información sobre los indicadores de evaluación de los algoritmos de procesamiento de imágenes y detección de objetos.

Los parámetros que se utilizan comúnmente para medir la eficacia y eficiencia de un detector son: el verdadero positivo (TP), el verdadero negativo (TN), el falso positivo (FP) y el falso negativo.

El concepto positivo y negativo se refiere básicamente a la presencia o no del recuadro de detección en la imagen de estudio, y el verdadero y el falso hace referencia al cumplimiento o no de la detección.

Para explicar mejor estos conceptos se puede analizar la figura 88, en este caso re relaciona la detección efectiva de la identificación de una señal de tránsito.

Figura 89. Ejemplo de detección de una señal de tránsito y su determinación del caso de efectividad del algoritmo

(Fuente: Gómez 2015)

Para calcular estos parámetros que permiten la evaluación de la eficacia del detector, se utilizan las siguientes ecuaciones:

$$P = \frac{\#\ Detecciones\ Verdaderas}{\#\ Detecciones} = \frac{TP}{TP + FP}$$

Donde P es la precisión y mide la probabilidad de que una detección sea verdadera lo ideal es que P=1

$$R = \frac{\#\ Detecciones\ Verdaderas}{\#\ Imágenes\ Positivas} = \frac{TP}{TP + FN}$$

En este caso R es la sensibilidad del detector y mide la probabilidad de que un objeto que esté realmente en la imagen sea detectado, igualmente, lo ideal es que R=1.

De la misma forma, para evaluar la eficacia del detector teniendo en cuenta la localización de la detección, se utilizan las siguientes ecuaciones teniendo en cuenta las regiones de intersección entre el recuadro de entrenamiento y el recuadro generado de la detección.

Esta situación se puede ilustrar en la figura 89.

Figura 90. Imagen con el recuadro de entrenamiento y el recuadro de detección

(Fuente: Gómez 2015)

Para calcular la precisión teniendo en cuenta la intersección de las áreas de los recuadros de entrenamiento y detección, se aplica la siguiente ecuación:

Para este caso la D, hace referencia a la lista de rectángulos de las detecciones y la G, hace referencia a la lista de rectángulos de las imágenes positivas o recuadros de entrenamiento.

$$P(G,D) = \frac{\sum_{j=1}^{|D|} BestMatch\ (D_j)}{|D|}$$

$$BestMatch(D_j) = max_i \frac{Area(G_i \cap D_j)}{Area(G_i \cap D_j)}$$

Y para calcular el recall o la sensibilidad se usan las siguientes ecuaciones, teniendo en cuenta la intersección de las áreas de los recuadros mencionados anteriormente.

$$R(G,D) = \frac{\sum_{j=1}^{|G|} BestMatch\ (G_j)}{|G|}$$

$$BestMatch(G_j) = max_i \frac{Area(G_i \cap D_j)}{Area(G_i \cap D_j)}$$

La valoración de la precisión y la sensibilidad se hace teniendo en cuenta intervalos óptimo, como se puede apreciar en la figura 90.

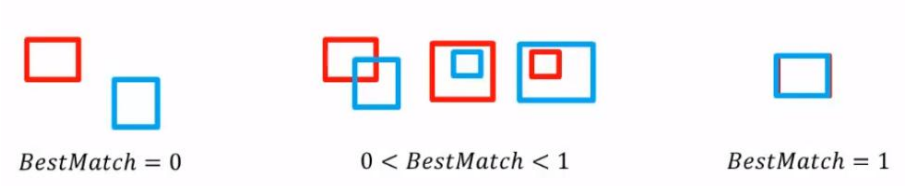

Figura 91. Intervalos de medición para la evaluación de los detectores

(Fuente: Gómez 2015)

En la figura 91, se presentan los casos extremos que pueden existir en la medición de la precisión y la sensibilidad de un detector. Hay que tener en cuenta que no por tener una alta sensibilidad y una alta precisión es el mejor detector, ya que un elemento de medición puede afectar al otro, por ende se hace necesario buscar valores intermedio en estos parámetros que puedan conllevar a seleccionar un detector eficiente.

Figura 92. Casos extremos de detección

(Fuente: Gómez 2015)

En este sentido, se utiliza una métrica que une a los dos indicadores de precisión y sensibilidad y se le denomina Medida F_1, en donde básicamente la ecuación es una relación de la precisión y la sensibilidad del detector.

$$F_1 = \frac{2PR}{P + R}$$

Teniendo en cuenta estos parámetros para medir la eficacia del detector, en el código en Matlab® que se mostró en la figura 87, hay unas líneas que permiten calcular de forma instantánea estos indicadores o parámetros de medición del detector, así que además de que el código generará el .XML, que se utilizará en Python, también dará un diagnóstico de la eficacia del detector que piensa utilizar.

4. PRUEBAS Y VERIFICACIONES

En este apartado haremos la verificación con imágenes tomadas del cultivo y verificaremos su comportamiento respecto a distancia de aplicación y robustez a la hora de identificación.

Los dos algoritmos que se escogieron para hacer las verificaciones y validaciones finales son: uno basado en la fusión del programa de identificación de figuras geométricas semicirculares con el de selección de colores en los niveles de RGB y uno de visión artificial basado en clasificadores en cascada tipo Haar y Hog usados ampliamente en la identificación de rostros y reconocimiento de objetos en imágenes respectivamente.

El procedimiento para la verificación y validación se realizará de la siguiente forma: se hacen 5 corridas, cada una con diferentes imágenes características seleccionadas, que se tomaron en el cultivo real, a cada una de esas corridas se le determina el porcentaje de detección, colocando la imagen a diferentes distancias iniciando a 10 cm, y comenzado a aumentar en 10 cm hasta llegar a 200 cm o 2 m en la mayoría de los casos. En cada una de estas distancias, se toma el tiempo de detección durante 2 minutos, para calcular de esta forma el % de detección en cada distancia.

Inicialmente, comenzaremos a validar el código desarrollado con base a figuras geométricas semicirculares y detección de colores en niveles RGB, en la figura 92, se muestra las imágenes del proceso de verificación del código con las imágenes mencionadas anteriormente.

En esta figura, cada cuadro hace referencia a cada imagen seleccionada para realizar los ensayos y cada cuadro hace referencia de las distancias a la cual se tomó la imagen.

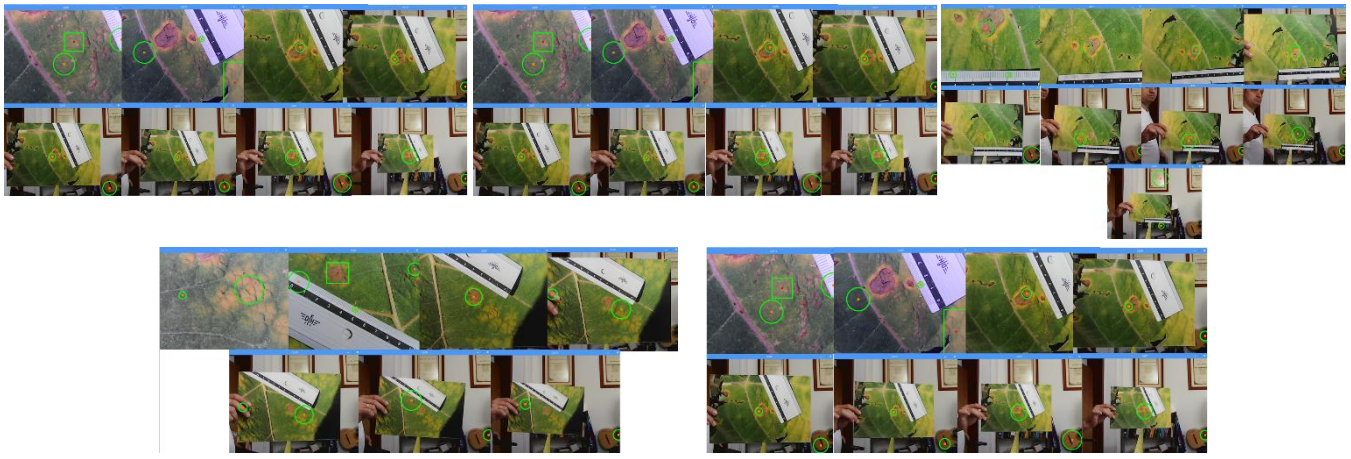

Figura 93. Proceso de validación y pruebas de detección del algoritmo de segmentación

(Fuente: Propia)

Con los datos cuantitativos arrojados en estas pruebas procedimos a graficar las detecciones obtenidas, en la figura 93 se muestra la gráfica de estos valores.

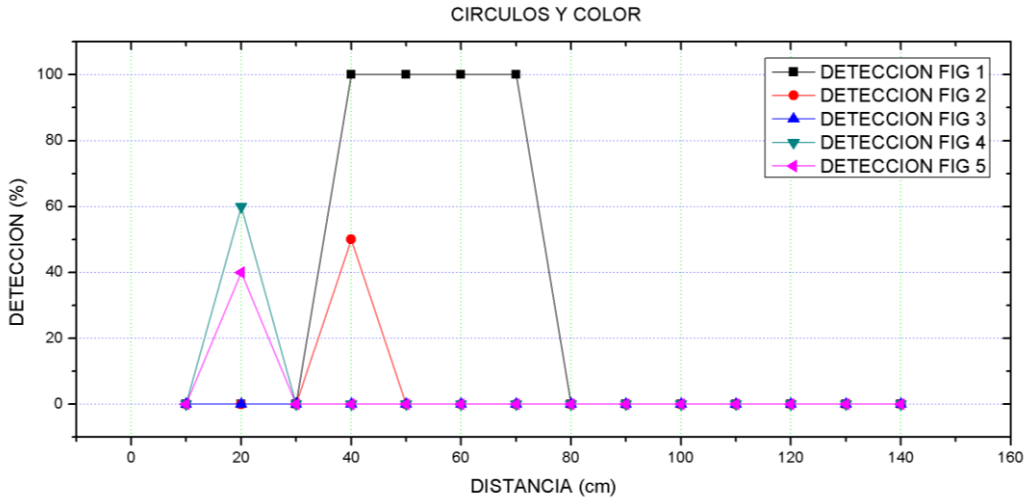

Figura 94. Datos de detección del algoritmo de segmentación

(Fuente: Propia)

A continuación, procedimos a realizar la validación en el ambiente simulado de los algoritmos desarrollados bajo el entrenamiento y despliegue usando clasificadores en cascada utilizando descriptores HOG, en la figura 94, se presenta el trabajo experimental que se desarrolló bajo las condiciones simuladas de detección.

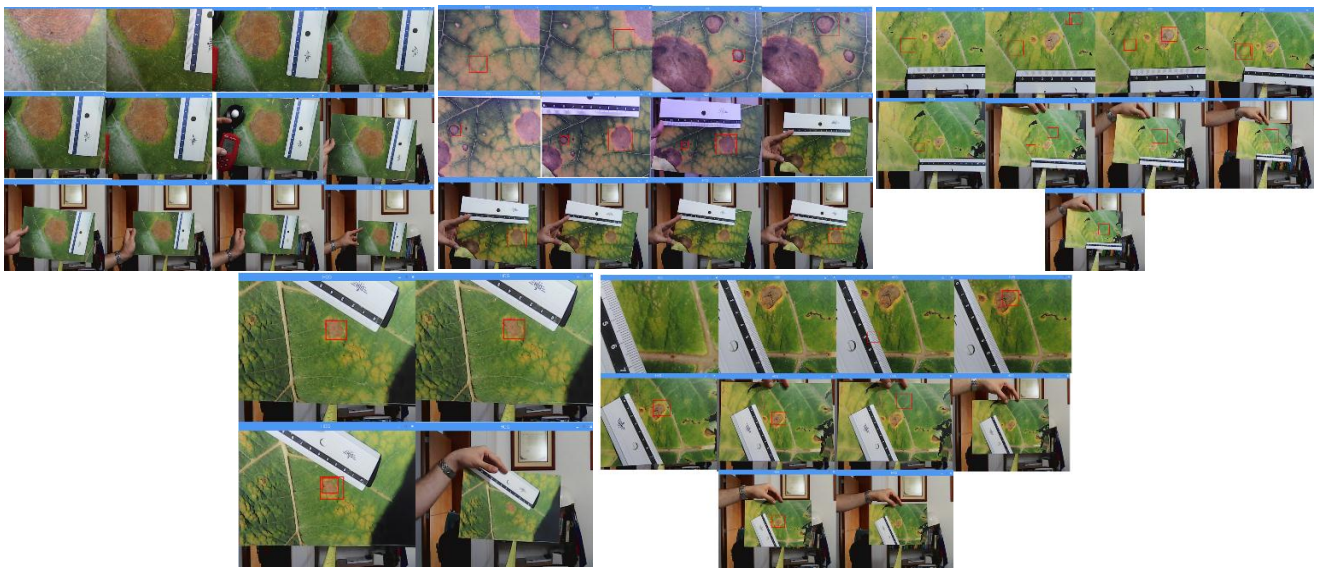

Figura 95. Proceso de validación y pruebas del algoritmo de detección Adaboost con descriptores HOG

(Fuente: Propia)

Con base a estas pruebas, se procedió a reunir los datos obtenidos referentes al porcentaje de detección que arrojó este algoritmo, es de anotar, que el entrenamiento de este se realizó con el software Matlab® con una base de entrenamiento de 196 imágenes positivas y 392 imágenes negativas, a través de esta información el entrenamiento se ejecutó en 7 etapas, las cuales, arrojó el archivo .xml que se aplicó.

En la figura 95, se muestra la gráfica obtenida de los datos de la prueba que se realizó en el ambiente simulado en las instalaciones de Tecnoparque Nodo Medellín, en esta figura se compara el porcentaje de detección del algoritmo con cada imagen de estudio, en él se puede ver que en algunas imágenes el algoritmo funciona mejor que en otras.

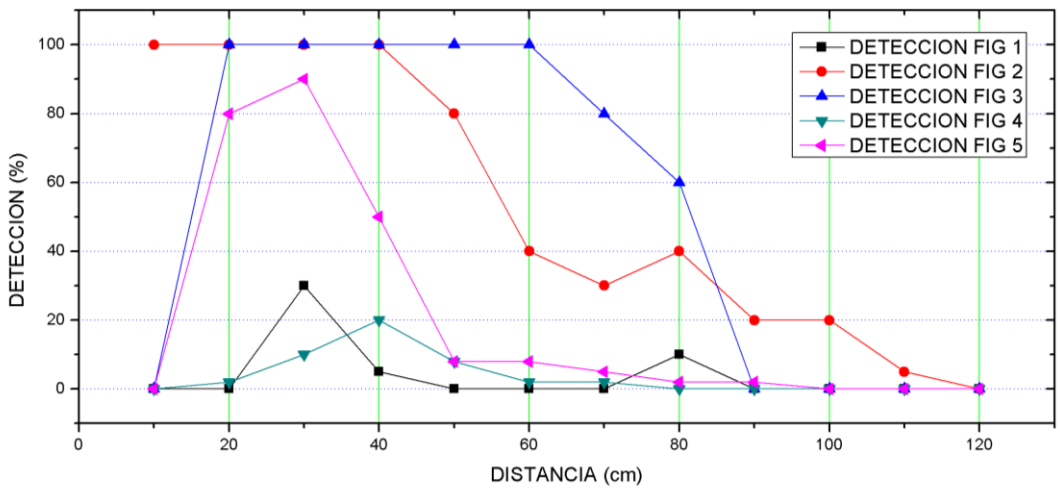

Figura 96. Datos de detección del algoritmo Adaboost usando descriptores HOG

(Fuente: Propia)

Igualmente, se procedió a realizar el trabajo de prueba con el algoritmo realizado con filtros Haar, para el desarrollo de este algoritmo se optó por trabajar la metodología dispuesta por [73], al tener ya el archivo .xml , se ingresa a un código de Python en donde puede ser ejecutado perfectamente por un open hardware como la Raspberry Pi.

En la figura 96, se muestra la detección que se tuvo en la prueba de ambiente, en donde se puede ver a groso modo el comportamiento de detección, en ella se puede observar que a una distancia muy cercana (10 cm) del objeto a la cámara no hay una buena detección.

Figura 97. Proceso de validación y pruebas del algoritmo Adaboost usando filtros tipo Haar (Fuente: Propia)

Con los datos resultados de este trabajo experimental en el ambiente simulado se procedió a graficar el porcentaje de detección del algoritmo, por cada imagen en función de la distancia, estos comportamientos se ilustran en la figura 97.

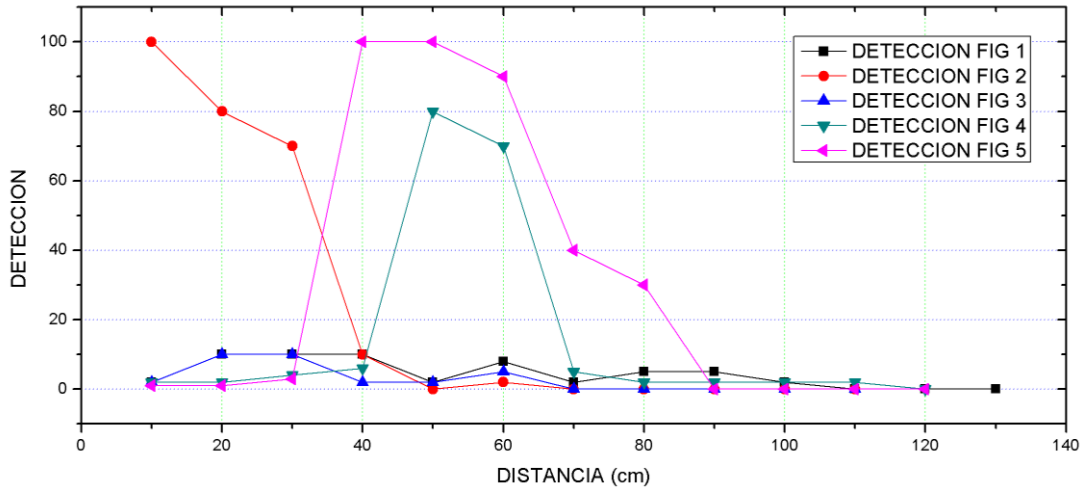

Figura 98. Figura 98. Gráfica del porcentaje de detección del algoritmo Adaboost usando filtros tipo Haar

(Fuente: Propia)

Teniendo los valores resultados de los ensayos de la aplicación de los códigos, al caso de estudio, se muestra en la figura 98, la comparación de los valores obtenidos, para ilustrar mejor, la efectividad en el ambiente simulado de estos algoritmos.

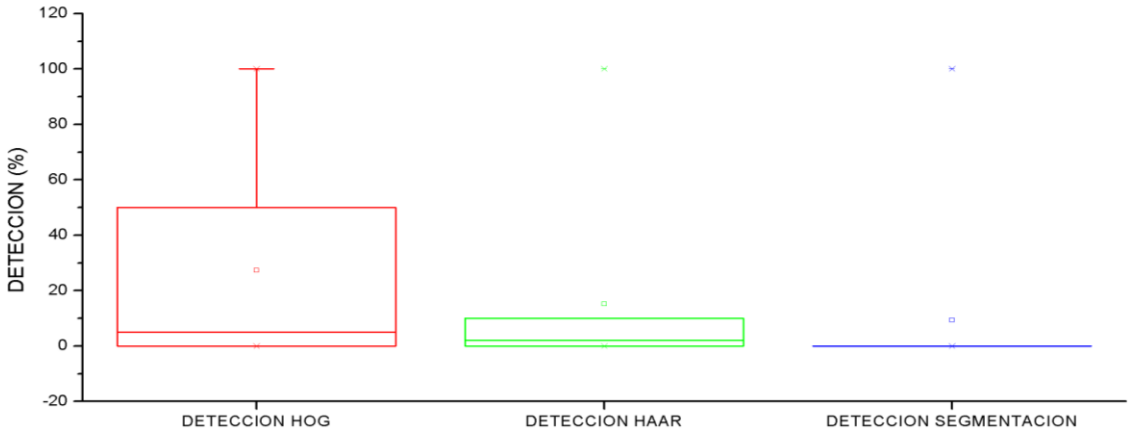

Figura 99. Comparación del porcentaje de detección de los tres algoritmos estudiados

(Fuente: Propia)

En esta figura se muestra que el algoritmo de detección que tuvo el mejor rendimiento, fue el que se realizó con descriptores HOG, ya que a nivel más básico es el que ha proporcionado los mejores resultados y sirve actualmente como referente en los sistema de detección [74]. Seguidamente, el que obtuvo el mejor rendimiento fue el detector con filtros Haar, el cual, en la literatura ha sido también altamente usado. No obstante, se puede evidenciar una gran diferencia en el comportamiento, y en el último puesto, está el algoritmo basado en procesos de segmentación, este método presenta múltiples inconvenientes y es muy poco robusto a cambios de iluminación, por lo que no se recomienda su uso para aplicaciones en exteriores. Todas estas gráficas, tanto las que representan el comportamiento del algoritmo en función de la distancia como la de caja y bigotes fueron realizadas en Origin®.

Así mismo, se realizó la verificación y pruebas en Matlab con el fin de terminar los valores de los descriptores que nos permitiera determinar la efectividad del algoritmo hibrido de visión artificial e inteligencia artificial, para realizar esta acción se procedió a ejecutar en dicho programa los códigos que se encuentran en las figuras 85 y 87, estos códigos generan las siguientes salidas.

El primer código (Figura 85), genera una visualización de la imagen analizada con la detección realizada del algoritmo, como se ve en la figura 99.

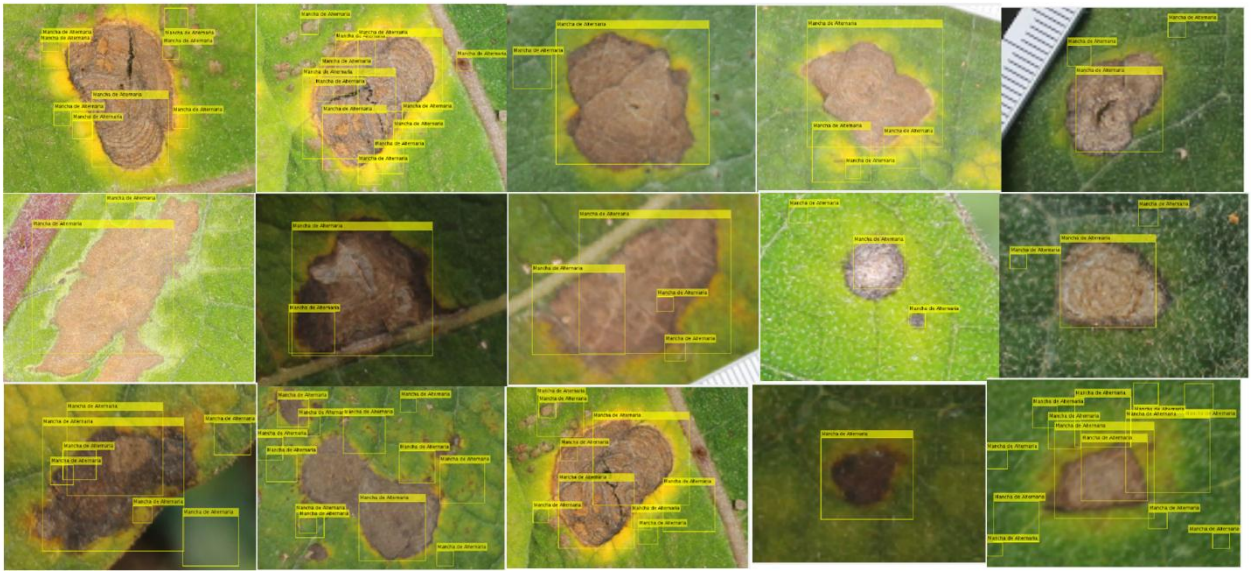

Figura 100. Imagen de las detecciones realizadas por el código HOG en Matlab

(Fuente: Propia)

Como se puede apreciar en la figura 99, se encuentran muchos falsos positivos y falsos negativos, sin embargo, el algoritmo si está detectando la presencia del objeto.

En la figura 100, se muestra el recuadro de entrenamiento (azul) y el recuadro de detección (rojo) en algunas imágenes positivas.

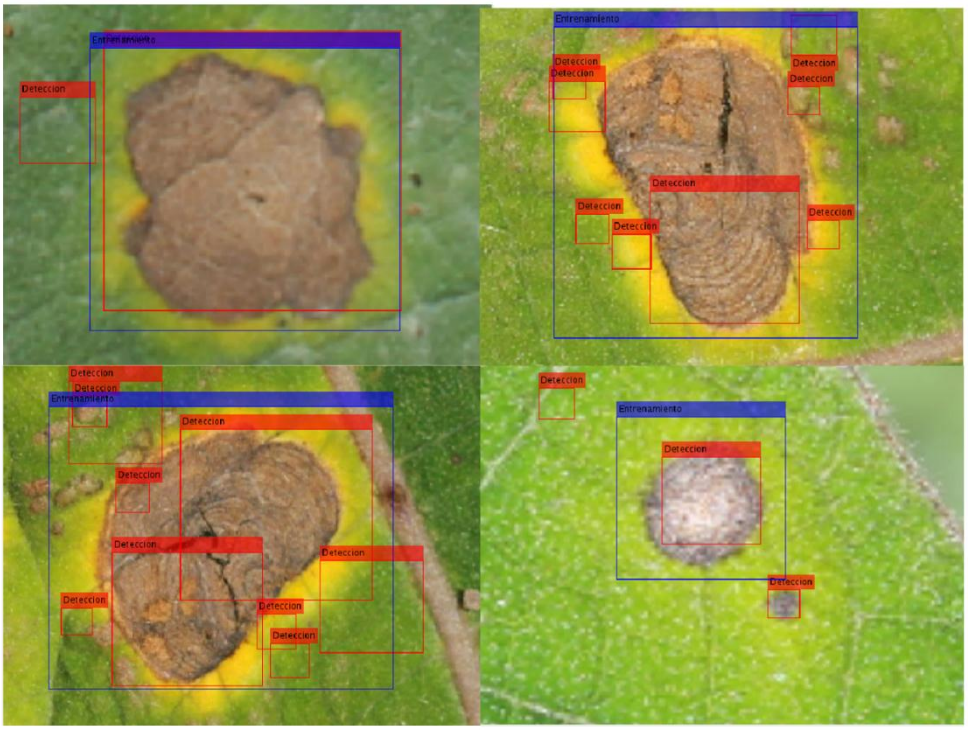

Figura 101. Imágenes de la comparación de la detección y el entrenamiento del algoritmo en Matlab

(Fuente: Propia)

El código de la figura 101, nos da los resultados cuantitativos respecto a la precisión y sensibilidad del algoritmo.

```
%Desempeño con los objetos de entrenamiento
Dtrain=detectandoSobreG(detectorName,Gtrain, false);
[F1_train, P, R] = evaluandoDetector(Gtrain,Dtrain)

% Validación con otras imagenes
load('positivastest.mat');
Dtest = detectandoSobreG(detectorName,positivasvalidacion2, false);
[F1_test, P, R] = evaluandoDetector(positivasvalidacion2, Dtest)

F1_train =
    0.1630
P =
    0.0999
R =
    0.4436
F1_test =
    0.1643
P =
    0.1012
R =
    0.4349
```

Figura 102. Resultados cuantitativos de las pruebas y validación del algoritmo en Matlab

(Fuente: Propia)

Como se puede corroborar tanto en el resultado visual como en el resultado cuantitativo, el algoritmo, con las imágenes positivas y negativas trabajadas, presenta un valor F1 bajo, lo que muestra que este algoritmo, según los valores de P y R, tiene una precisión baja y una sensibilidad relativamente alta, lo que no permite que sea un buen detector en su clase, no obstante, presenta un buen comportamiento de detección respecto al algoritmo Haar y al algoritmo de segmentación estudiados en este trabajo, bajo las condiciones de entrenamiento establecidas.

5. CONCLUSIONES

El desarrollo de técnicas de visión artificial está sujeto al tipo de aplicación, ya sea en entornos acondicionados diseñados específicamente para esta actividad o para condiciones externas, debido a que un factor importante en la visión artificial es la iluminación que se tenga para que los detectores puedan identificar de forma clara el objeto a detectar.

Las técnicas de segmentación ya sea por umbralización u otro mecanismo son aplicables siempre y cuando se tenga un manejo adecuado de la fuente de luz que permita aplicar de forma correcta los filtros y no se tengan objetos errados a causas de sombras o brillos.

El procesamiento de imágenes e identificación de objetos es mejor realizarlo en un pc con características comunes, ya que los procesadores que tienen los sistemas embebidos no soportan el procesamiento de imágenes en línea de forma eficiente si se quiere hacer una detección de objetos en tiempo real, por ende, se consigue en la literatura muchos sistemas de procesamiento de imágenes off-line.

El procesamiento de imágenes también depende de gran medida de los periféricos o hardware que se tengan para el desarrollo del sistema de visión artificial, por ende, entre más resolución tenga el sensor de imagen o más capacidad de emitir los *frames* por segundo, mejor será la detección de los objetos.

Los algoritmos de detección de figuras geométricas y los desarrollados a partir de detectores en cascada presentan más robustez que la segmentación de colores, debido a que la iluminancia evidenciada en los trabajos experimentales le genera mucho ruido en la detección a pesar de que se maneje una tolerancia de RGB entre +/- 20 y +/- 40.

A pesar que el algoritmo de inteligencia artificial de detectores en cascada usando Adabost con filtros Haar es muy usado en la detección de rostros presentó menor eficacia en comparación con el algoritmo Adabost desarrollados con descriptores de Histogramas de gradientes orientados (HOG), esto se debe a que las manchas originadas por el hongo no tiene una forma regular fácil

de identificar como es el caso de los rostros que tienen unos parámetros definidos cualquiera que sea la cara que se quiera detectar.

El proceso de entrenamiento y la base de datos dispuesta para tal fin, es fundamental en el desarrollo de un algoritmo de inteligencia artificial, por ende, si se quiere tener resultados de detección con una valor de eficacia o F1 elevados o aceptables, se debe contar primero, con una buena cantidad de imágenes positivas y negativas, sabiendo que el número de imágenes negativas debe duplicar el número de positivas y segundo realizar un proceso de escalamiento de imágenes y de selección de región de interés (ROI) adecuado.

Se pudo determinar que con la base de entrenamiento entre 200 a 400 imágenes se puede detectar la mancha de Alternaria a una distancia entre 40 a 60 cm, usando el algoritmo de Adaboost con un filtro HOG.

El presente trabajo de investigación propuso el desarrollo de un sistema de detección y reconocimiento de patrones, a través de la visión e inteligencia artificial usando el algoritmo *Adaboos*t, capaz de identificar la señal de la presencia de un hongo en el cultivo del lulo desarrollado con hardware y software libre, y que puede ser empleado en vehículos aéreos no tripulados, no obstante, de este trabajo se pudo determinar que se requiere una muy buena base de entrenamiento de imágenes con la patología que se quiere identificar y un sensor de imagen de buena calidad o resolución, ya que este es determinante para ampliar la distancia que se debe tener desde el drone a la hoja, igualmente, en posteriores trabajos se deberán estudiar otras técnicas de inteligencia artificial como redes neuronales o lógica difusa, orientadas a la detección de objetos o reconocimiento de patrones.

REFERENCIAS BIBLIOGRAFICAS

[1] J. De Baerdemaeker, *Precision Agriculture Technology and Robotics for Good Agricultural Practices*, vol. 46, no. 4. IFAC, 2013.

[2] P. Tamayo, R. Navarro, and M. C. de la Rotta, “Enfermedades del cultivo del lulo en Colombia,” *Boletín Técnico 9 - CORPOICA*, 2001.

[3] et all Rogério dos Santos Alves; Alex Soares de Souza, “Manejo fitosanitario del cultivo del lulo (Solanum quitoense Lam)- Medidas para la temporada invernal,” *Igarss 2014*, no. 1, pp. 1–5, 2014.

[4] D. MinAgricultura, *El Cultivo del Lulo (Solanum quitoense) y los efectos del fenómeno del niño en la producción .* 2015.

[5] M. M. González Ponce, “Visión por Computador para UAS,” Universidad Politécnica de Madrid, 2012.

[6] J. Torres-Sánnchez, J. M. Peña, A. I. de Castro, and F. López-Granados, “Multi-temporal mapping of the vegetation fraction in early-season wheat fields using images from UAV,” *Comput. Electron. Agric.*, vol. 103, pp. 104–113, 2014.

[7] D. Doering *et al.*, “MDE-based Development of a Multispectral Camera for Precision Agriculture,” *IFAC-PapersOnLine*, vol. 49, no. 30, pp. 24–29, 2016.

[8] D. Doering *et al.*, *Design and optimization of a heterogeneous platform for multiple UAV use in precision agriculture applications*, vol. 19, no. 3. IFAC, 2014.

[9] J. Senthilnath, A. Dokania, M. Kandukuri, R. K.N., G. Anand, and S. N. Omkar, “Detection of tomatoes using spectral-spatial methods in remotely sensed RGB images captured by UAV,” *Biosyst. Eng.*, vol. 146, pp. 16–32, 2016.

[10] E. Chartuni, F. De Assis De Carvalho, D. Marçal, and E. Ruz, “Perspectivas Agricultura de precisión Nuevas herramientas para mejorar la gestión tecnológica en la empresa agropecuaria,” 2007.

[11] N. Zhang, M. Wang, and N. Wang, “Precision agriculture—a worldwide overview,” *Comput. Electron. Agric.*, vol. 36, no. 2–3, pp. 113–132, 2002.

[12] Procisur, *Manual de agricultura de precisión*. 2014.

[13] D. F. Jaramillo and S. Sadeghian, “Spatial Variability of Bases in an Andisol of the Colombian Central Coffee Zone,” *Boletín Ciencias la Tierra*, no. 33, pp. 111–124, 2013.

[14] J. D. Muñoz, L. J. Martinez, and R. Giraldo, “Variabilidad espacial de propiedades edáficas y su relación con el rendimiento en un cultivo de papa,” *Agron. Colomb.*, vol. 24, no. 2, pp. 355–366, 2006.

[15] J. Rodríguez, A. M. González, F. Rodrigo, and L. Guerrero, “Fertilización por sitio específico en un cultivo de maíz (Zea mays L .) en la Sabana de Bogotá,” *Agron. Colomb.*, vol. 26, no. 2, pp. 308–321, 2008.

[16] C. andrés Garzón Gutiérrez, C. A. Cortés, and J. H. Camacho Tamayo, “Variabilidad espacial de algunas propiedades quimicas en un entisol,” *Rev. UDCA Actual. Divulg. Científica*, vol. 13, no. 1, pp. 87–95, 2010.

[17] R. M. Castellanos and M. Morales Pérez, “Análisis crítico sobrela conceptualización de la agricultura de precisión,” *Cienc. en su PC*, vol. 2, pp. 23–33, 2016.

[18] F. Capraro, S. Tosetti, and F. Vita Serman, “Laboratorio Virtual y Remoto para Simular, Monitorizar y Controlar un Sistema de Riego por Goteo en Olivos,” *Rev. Iberoam. Automática e Informática Ind. RIAI*, vol. 7, no. 1, pp. 73–84, 2010.

[19] Ó. Orozco Sarasti and G. Llano Ramírez, “Sistemas de Información enfocados en tecnologías de agricultura de precisión y aplicables a la caña de azúcar, una revisión,” *Rev. Ing. Univ. Medellín, ISSN 1692-3324, Vol. 15, N^{o}. 28, 2016*, vol. 15, no. 28, p. 6, 2016.

[20] Congreso de Colombia, “Ley 1753 de 2015 (junio 9) por la cual se expide el Plan Nacional de Desarrollo 2014-2018 ;Todos por un nuevo país,” vol. 2015, no. junio 9, p. 104, 2015.

[21] CONPES, “Conpes 3582,” p. 68, 2009.

[22] Asme V&V 10.1-2012, “An Illustration of the Concepts of Verification and Validation in Computational Solid Mechanics,” 2012.

[23] P. Cruz, K. Acosta, J. R. Cure, and D. Rodriguez, “Desarrollo y fenologia del lulo solanum quitoense var. septentrionale bajo polisombra desde siembra hasta primera fructificacion.,” *Agron. Colomb.*, vol. 25, pp. 288–298, 2007.

[24] C. A. Quinchia, F., Cabrera, “Manual tecnico del cultivo de lulo (Solanum quitoense L) en el departamento del Huila,” p. 34, 2006.

[25] P. C. Robert, “Precision Agriculture: an Information Revolution in Agriculture,” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 1999.

[26] R. Bongiovanni, E. Chartuni Mantovani, S. Best, and Á. Roel, “Agricultura de Presición: Integando Conocimentos para una Agricultura Moderna y Sustentable,” p. 244, 2006.

[27] B. S. Faiçal *et al.*, “An adaptive approach for UAV-based pesticide spraying in dynamic environments,” *Comput. Electron. Agric.*, vol. 138, pp. 210–223, 2017.

[28] R. W. Coates, M. J. Delwiche, A. Broad, and M. Holler, “Wireless sensor network with irrigation valve control,” *Comput. Electron. Agric.*, vol. 96, pp. 13–22, 2013.

[29] J. Gimenez, D. Herrera, S. Tosetti, and R. Carelli, “Optimization methodology to fruit grove mapping in precision agriculture,” *Comput. Electron. Agric.*, vol. 116, pp. 88–100, 2015.

[30] A. G. Mohapatra and S. K. Lenka, “Neural Network Pattern Classification and Weather Dependent Fuzzy Logic Model for Irrigation Control in WSN Based Precision Agriculture,” *Phys. Procedia*, vol. 78, no. December 2015, pp. 499–506, 2016.

[31] J. Rasmussen, G. Ntakos, J. Nielsen, J. Svensgaard, R. N. Poulsen, and S. Christensen, “Are vegetation indices derived from consumer-grade cameras mounted on UAVs sufficiently reliable for assessing experimental plots?,” *Eur. J. Agron.*, vol. 74, pp. 75–92, 2016.

[32] K. Ribeiro-Gomes, D. Hernandez-Lopez, R. Ballesteros, and M. A. Moreno, “Approximate georeferencing and automatic blurred image detection to reduce the costs of UAV use in environmental and agricultural applications,” *Biosyst. Eng.*, vol. 151, pp. 308–327, 2016.

[33] P. J. Zarco-Tejada, R. Diaz-Varela, V. Angileri, and P. Loudjani, “Tree height quantification using very high resolution imagery acquired from an unmanned aerial vehicle (UAV) and automatic 3D photo-reconstruction methods,” *Eur. J. Agron.*, vol. 55,

pp. 89–99, 2014.

[34] J. Gago *et al.*, “UAVs challenge to assess water stress for sustainable agriculture,” *Agric. Water Manag.*, vol. 153, pp. 9–19, 2015.

[35] L. G. Santesteban, S. F. Di Gennaro, A. Herrero-Langreo, C. Miranda, J. B. Royo, and A. Matese, “High-resolution UAV-based thermal imaging to estimate the instantaneous and seasonal variability of plant water status within a vineyard,” *Agric. Water Manag.*, vol. 183, pp. 49–59, 2017.

[36] J. Polo, G. Hornero, C. Duijneveld, A. García, and O. Casas, “Design of a low-cost Wireless Sensor Network with UAV mobile node for agricultural applications,” *Comput. Electron. Agric.*, vol. 119, pp. 19–32, 2015.

[37] J. Torres-Sánchez, F. López-Granados, and J. M. Peña, “An automatic object-based method for optimal thresholding in UAV images: Application for vegetation detection in herbaceous crops,” *Comput. Electron. Agric.*, vol. 114, pp. 43–52, 2015.

[38] X. P. Burgos-Artizzu, A. Ribeiro, and M. De Santos, “Controlador Borroso Multivariable para el Ajuste de Tratamientos en Agricultura de Precisión,” *Rev. Iberoam. Automática e Informática Ind. RIAI*, vol. 4, no. 2, pp. 64–71, 2007.

[39] K. N. V. P. S. Rajesh and R. Dhuli, “Classification of imbalanced ECG beats using re-sampling techniques and AdaBoost ensemble classifier,” *Biomed. Signal Process. Control*, vol. 41, pp. 242–254, 2018.

[40] D. Lozano, C. Arranja, M. Rijo, and L. Mateos, “Simulation of automatic control of an irrigation canal,” *Agric. Water Manag.*, vol. 97, no. 1, pp. 91–100, 2010.

[41] C. Smith and A. Corripio, *Control Automatico de Procesos teorica y práctica*, 2a ed. 1991.

[42] P. Ponce Cruz, *Inteligencia artificial con aplicaciones a la ingenieria*, 1a ed. 2010.

[43] M. Santos, “Un enfoque aplicado del control inteligente,” *RIAI - Rev. Iberoam. Autom. e Inform. Ind.*, vol. 8, no. 4, pp. 283–296, 2011.

[44] P. Cano Marchal, J. Gómez Ortega, D. Aguilera Puerto, and J. Gámez García, “Situación actual y perspectivas futuras del control del proceso de elaboración del aceite de oliva virgen,” *Rev. Iberoam. Automática e Informática Ind. RIAI*, vol. 8, no. 3, pp. 258–269, 2011.

[45] A. K. Torres Galindo, “Development of a multispectral system for precision agriculture applications using embedded devices,” *Sist. y Telemática*, vol. 13, no. 33, p. 27, 2015.

[46] A. Jimenénez, D. Ravelo, and J. Gómez, “Sistema de adquisición, almacenamiento y análisis de información fenológica para el manejo de plagas y enfermedades de un duraznero mediante tecnologías de agricultura de precisión,” *Tecnura*, vol. 14, pp. 41–51, 2010.

[47] G. D. De León Mata, A. Pinedo Álvarez, and J. H. Martínez Guerrero, “Aplicación de sensores remotos en el análisis de la fragmentación del paisaje en Cuchillas de la Zarca, México,” *Investig. Geogr.*, vol. 84, no. 84, pp. 42–53, 2014.

[48] F. J. Espinosa-Faller and G. E. Rendón-Rodríguez, “A ZigBee Wireless Sensor Network for Monitoring an Aquaculture Recirculating System,” *J. Appl. Res. Technol.*, vol. 10, no. 3, 2012.

[49] R. S. Sinha, Y. Wei, and S.-H. Hwang, “A survey on LPWA technology: LoRa and NB-IoT,” *ICT Express*, vol. 3, no. 1, pp. 14–21, 2017.

[50] G. Viera-Maza, “Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi para la clasificación del cacao,” Universidad de Piura, 2017.

[51] BIOS, “Centro de Bioinformática y biología computacional,” *Curso de visualización de datos y simulación en salud*, 2018. [Online]. Available: https://formacion.bios.co/.

[52] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 2nd ed. New Jersey, 2008.

[53] D. Silva Montemayor, “Estudio de viabilidad de un sistema basado en Raspberry Pi para aplicaciones de Inspección Industrial por Visión Artificial,” Universidad de Oviedo, 2015.

[54] C. Pérez González, “Deteccion y seguimiento de objetos por colores en una plataforma raspberry pi,” Universidad Politécnica de Madrid, 2016.

[55] D. Betancourt Gualteros, “Sistema De Visión Por Computador Para Detectar Hierba No Deseada En Prototipo De Cultivo De Frijol Usando Ambiente Controlado,” Universidad Católica de Colombia, 2014.

[56] E. Á. Sobrado Malpartida, “Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot,” Pontificia Universidad Católica del Perú, 2003.

[57] G. van Rossum, “El tutorial de Python,” *Python*, 2009. [Online]. Available: http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf.

[58] G. Bradski and A. Kaehler, *Projection and 3D Vision*. 2008.

[59] G. Bradski and A. Kaehler, *Learning OpenCV Computer Vision with the OpenCV Library*, Primera. O’Reilly, 2008.

[60] V. A. B. Meneses, J. M. Téllez, and D. F. A. Velasquez, “Uso De Drones Para El Analisis De Imágenes Multiespectrales En Agricultura De Precisión,” *@limentech, Cienc. y Tecnol. Aliment.*, vol. 13, no. 1, pp. 28–40, 2015.

[61] C. R. Jiménez Tenorio, “Construcción de un ordenador electrónico vehicular con sistema de seguridad y GPS utilizando RASPBERRY PY y hardware libre,” Universidad Técnica de Ambato, 2015.

[62] D. Garcia Gadea, “Sistema autónomo y de bajo coste para reconocimiento de códigos QR,” Universidad de Alicante, 2016.

[63] Raspberry Pi foundation, “Raspberry PI,” 2018. [Online]. Available: https://www.raspberrypi.org.

[64] O. E. Gualdrón, O. M. Duque Suaréz, and M. A. Chacón Rojas, “Diseño de un sistema de reconocimiento de rostros mediante la hibridación de técnicas de reconocimiento de patrones, visión artificial e IA, enfocado a la seguridad e interacción robótica social,” *Univiersidad La Rioja*, vol. 6, p. 13, 2013.

[65] CUCOPC, “Raspberry pi – Reconocimiento Facial OpenCV Python #2,” *CUCOPC*, 2016. [Online]. Available: https://cucopc.es/2016/06/09/raspberry-pi-reconocimiento-facial-opencv/.

[66] R. Lienhart, A. Kuranov, and V. Pisarevsky, “Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection,” pp. 297–304, 2003.

[67] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Proc. 2001 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition. CVPR 2001*, vol. 1, p. I-511-I-518, 2001.

[68] S. Rivera Castaño, D. J. Salas Álvarez, and R. J. Montes Rodríguez, “Identificación de la punta de los dedos de la mano en un plano 2D basado en Kinect,” *Gerenc. en Tecnol.*

Informática, vol. 12, pp. 57–65, 2013.

[69] OpenCV, “OpenCV,” 2018. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html .

[70] L. M. Guevara, J. D. Echeverry, and W. Ardila, “Detección de rostros en imágenes digitales usando clasificadores en cascada,” *Sci. Tech.*, vol. XIV, no. 38, pp. 1–6, 2008.

[71] E. E. Avila Romero, “Seguimiento de personas basado en los descriptores HOG,” 2011.

[72] Y. Freund and R. E. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” *J. Comput. Sist. Sci.*, vol. 139, pp. 23–37, 1995.

[73] M. Rezaei, “Creating a Cascade of Haar-Like Classifiers : Step by Step,” pp. 1–8, 2014.

[74] C. A. Luna, C. Losada-Gutierrez, D. Fuentes-Jimenez, A. Fernandez-Rincon, M. Mazo, and J. Macias-Guarasa, “Robust people detection using depth information from an overhead Time-of-Flight camera,” *Expert Syst. Appl.*, vol. 71, pp. 240–256, 2017.