UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA DE ESTIMACIÓN DE PRODUCCIÓN DE CULTIVO DE MORA EMPLEANDO VISIÓN ARTIFICIAL Y MACHINE LEARNING

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Programación y redes

LÍNEA DE INVESTIGACIÓN: Tecnología de información y Sistemas de control

AUTOR: Fabricio Javier Aldaz Saca

Gilber Andrés Ibarra Rojano

TUTOR: Ing. Edgar Patricio Córdova Córdova, Mg

Ambato - Ecuador

febrero – 2025

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE ESTIMACIÓN DE PRODUCCIÓN DE CULTIVO DE MORA EMPLEANDO VISIÓN ARTIFICIAL Y MACHINE LEARNING, desarrollado bajo la modalidad Proyecto de Investigación por los señores Fabricio Javier Aldaz Saca y Gilber Andrés Ibarra Rojano, estudiantes de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que los estudiantes han sido tutorados durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, febrero 2025.

Firmado electrónicamente por:
EDGAR PATRICIO
CORDOVA CORDOVA

Ing. Edgar Patricio Córdova Córdova, Mg
TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA DE ESTIMACIÓN DE PRODUCCIÓN DE CULTIVO DE MORA EMPLEANDO VISIÓN ARTIFICIAL Y MACHINE LEARNING es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, febrero 2025.

Fabricio Javier Aldaz Saca
C.C. 1804756375
AUTOR

Gilber Andrés Ibarra Rojano
C.C. 1804746533
AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, febrero 2025.

Fabricio Javier Aldaz Saca

C.C. 1804756375

AUTOR

Gilber Andrés Ibarra Rojano

C.C. 1804746533

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Fabricio Javier Aldaz Saca estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE ESTIMACIÓN DE PRODUCCIÓN DE CULTIVO DE MORA EMPLEANDO VISIÓN ARTIFICIAL Y MACHINE LEARNING, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, febrero 2025.

Firmado electrónicamente por:
FRANKLIN OSWALDO
MAYORGA MAYORGA

Ing. Franklin Oswaldo Mayorga Mayorga, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
GUSTAVO DANILO
MACHADO MERINO

Ing. Gustavo Machado Merino
PROFESOR CALIFICADOR

Firmado electrónicamente por:
FABIAN RODRIGO
SALAZAR ESCOBAR

Ing. Fabian Salazar Escobar, PhD
PROFESOR CALIFICADOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Gilber Andrés Ibarra Rojano estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE ESTIMACIÓN DE PRODUCCIÓN DE CULTIVO DE MORA EMPLEANDO VISIÓN ARTIFICIAL Y MACHINE LEARNING, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, febrero 2025.

Firmado electrónicamente por:
FRANKLIN OSWALDO MAYORGA MAYORGA

Ing. Franklin Oswaldo Mayorga Mayorga, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
CARLOS DIEGO GORDON GALLEGOS

Ing. Carlos Gordon Gallegos, PhD
PROFESOR CALIFICADOR

Firmado electrónicamente por:
CLARA AUGUSTA SANCHEZ BENITEZ

Ing. Clara Sánchez Benítez, Mg
PROFESOR CALIFICADOR

DEDICATORIA

El presente proyecto de investigación lo dedico, en primer lugar, a Dios. Ha sido mi guía y fortaleza en cada momento de este proceso, iluminando mi camino en tiempos de incertidumbre y dándome el valor necesario para superar los retos. Su presencia ha sido el pilar fundamental en mi vida y en este logro académico, que solo ha sido posible gracias a su gracia y sabiduría infinita.

A mis queridos padres, cuyo amor incondicional, sacrificio y fe en mí han sido el motor que me ha impulsado a alcanzar este hito. En especial, a mi madre, por ser el ejemplo de lucha, dedicación y valentía que me ha inspirado a dar siempre lo mejor de mí. A mi padre, por enseñarme a mantenerme firme ante las adversidades y por ser mi apoyo constante en cada decisión importante.

A mi hermana, Flor quien ha sido mi inspiración diaria y motivo de esfuerzo constante. Sus palabras de aliento, compañía y amor incondicional han sido el combustible que me ha permitido seguir adelante, incluso en los momentos más difíciles.

Este proyecto no solo representa mi esfuerzo, sino el amor, apoyo y sacrificio de mi familia, quienes han estado conmigo en cada paso del camino. Este logro les pertenece tanto a ustedes como a mí.

Fabricio Aldaz

AGRADECIMIENTO

A Dios, por concederme la salud, la sabiduría y la fortaleza necesarias para culminar este importante logro académico. Su guía y presencia han sido mi mayor fuente de inspiración y motivación en cada paso.

A mis padres y hermana, por su constante apoyo, amor incondicional y confianza en mí. Sus palabras de aliento y sacrificios han sido fundamentales para alcanzar este objetivo.

A mis abuelitos, por ser un ejemplo de vida y por sus valiosos consejos que me han inspirado a esforzarme y superarme día a día.

A mis amigos Andrés, Christian, Henry, Alexander y Mary, cuya compañía y motivación han sido un soporte inquebrantable a lo largo de este camino. Su amistad ha sido una fuerza que me impulsó a no rendirme.

Al Ingeniero de Campo de Innova Campo, Paul Chicaiza, por su valiosa colaboración y aportes en el desarrollo de esta investigación. Su experiencia y conocimientos enriquecieron enormemente este proyecto.

A mi tutor de tesis, el Ingeniero Patricio Córdova, por el tiempo dedicado a guiarme, apoyarme y brindarme sus conocimientos. Su orientación y experiencia fueron pilares fundamentales para la culminación de este trabajo.

A todos ellos, mi más sincero agradecimiento.

Fabricio Aldaz

DEDICATORIA

El presente proyecto de investigación se la dedico mis queridos padres, quienes con su amor, esfuerzo y sacrificio han sido mi mayor fuente de fortaleza e inspiración. Su ejemplo me ha enseñado el valor del trabajo duro y la perseverancia.

A mi madre María, en especial, por su inagotable cariño, sus palabras de aliento en los momentos más difíciles y por ser mi guía en cada paso de este camino. Mamá, tus enseñanzas y apoyo incondicional han sido el motor que me ha impulsado a seguir adelante.

A mi familia, por estar siempre presente, por su comprensión y por brindarme el soporte necesario para alcanzar mis metas. Este logro no habría sido posible sin el amor y la unión que siempre me han regalado.

Con todo mi corazón, este trabajo es para ustedes.

Andrés Ibarra

AGRADECIMIENTO

En primer lugar, quiero expresar mi más profundo agradecimiento a Dios, quien con su infinita bondad y guía ha iluminado cada paso de este camino, brindándome fortaleza, sabiduría y perseverancia para llegar hasta aquí.

A mis queridos padres, por su amor incondicional, sacrificios y constante apoyo. Gracias por ser mi ejemplo, mi más sincero reconocimiento y gratitud por su dedicación, sus palabras de aliento y su fe en mí incluso en los momentos más difíciles. Sin su amor y confianza, este sueño no habría sido posible.

A mi familia, por estar siempre presente y por darme el ánimo necesario para continuar cuando las fuerzas parecían flaquear.

A mi tutor, Ing. Edgar Patricio Córdova, por su guía, paciencia y compromiso. Su experiencia y consejos han sido una pieza clave en el desarrollo de esta investigación.

A mi compañero de investigación y amigo Fabricio por su esfuerzo, dedicación y trabajo en equipo. Gracias por compartir ideas, retos y logros conmigo en esta travesía académica.

A mis amigos, por los momentos compartidos, las risas y el apoyo mutuo en cada etapa de este largo camino. Su compañía hizo que este camino fuera más llevadero y especial.

A todos ustedes, mi más sincero agradecimiento. Este logro no habría sido posible sin el apoyo, el amor y la fe que cada uno de ustedes ha depositado en mí.

Andrés Ibarra

ÍNDICE GENERAL DE CONTENIDOS

PORTADA ..... **i**

APROBACIÓN DEL TUTOR ..... **ii**

AUTORÍA ..... **iii**

DERECHOS DE AUTOR ..... **iv**

APROBACIÓN DEL TRIBUNAL DE GRADO ..... **vi**

DEDICATORIA ..... **vii**

AGRADECIMIENTO ..... **viii**

DEDICATORIA ..... **ix**

AGRADECIMIENTO ..... **x**

ÍNDICE GENERAL DE CONTENIDOS ..... **xi**

ÍNDICE DE TABLAS ..... **xv**

ÍNDICE DE FIGURAS ..... **xvii**

ÍNDICE DE ANEXOS ..... **xxii**

RESUMEN EJECUTIVO ..... **xxiii**

ABSTRACT ..... **xxiv**

CAPÍTULO I. MARCO TEÓRICO **1**

1.1 Tema de investigación 1

1.1.1 Planteamiento del problema 1

1.2 Antecedentes investigativos 3

1.3 Fundamentación teórica 6

1.3.1 Agricultura de Precisión 6

1.3.2 Tecnologías involucradas en la agricultura de precisión 7

1.3.3 Cultivo de Mora 8

1.3.4 Métodos para la producción de mora 12

1.3.5 Factores que afectan a la producción de mora 14

1.3.6 Inteligencia Artificial en la Agricultura 16

1.3.7 Métodos usados actualmente para estimar la cantidad de producción en la agricultura 19

1.3.8 Visión Artificial en el monitoreo de cultivos 20

1.3.9 Vehículos Aéreos no tripulados para la adquisición de imágenes 25

1.3.10 Planificación de trayectorias 29

1.3.11 Algoritmos de machine learning usados en la predicción de producción. 37

1.3.12 Modelos y plataformas de inteligencia artificial para la agricultura 44

1.3.13 Microordenador 47

1.3.14 Sistema de Gestión de Base de Datos (SGBD) 49

1.4 Objetivos 54

1.4.1 Objetivo general 54

1.4.2 Objetivos específicos 54

CAPÍTULO II. METODOLOGÍA 55

2.1 Materiales 55

2.2 Métodos 56

2.2.1 Modalidad de la investigación 56

2.2.2 Recolección de información 57

2.2.3 Procesamiento y análisis de datos 58

CAPÍTULO III. RESULTADOS Y DISCUSIÓN 59

3.1 Análisis y discusión de los resultados 59

3.2 Descripción del cultivo de mora a monitorear 59

3.3 Desarrollo general de la propuesta 61

3.3.1 Selección de Tecnología 61

3.3.2 Selección de software 63

3.3.3 Arquitectura del sistema 64

3.3.4 Diseño del sistema 71

3.3.5 Implementación del sistema 112

3.3.6 Resultados de las pruebas de funcionamiento 118

3.3.7 Requisitos mínimos de hardware 123

3.3.8 Requisitos mínimos de software 124

3.3.9 Costo computacional 124

3.3.10 Presupuesto 125

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES 127

4.1 Conclusiones 127

4.2 Recomendaciones 129

REFERENCIAS BIBLIOGRÁFICAS 131

ANEXOS 135

ÍNDICE DE TABLAS

Tabla 1. Tecnologías usadas comúnmente en la agricultura de precisión. 7

Tabla 2. Provincias con la mayor producción de mora en Ecuador. 8

Tabla 3. Descriptores Cuantitativos y cualitativos morfológicos. 12

Tabla 4. Características de las condiciones ambientales para el cultivo de mora. 13

Tabla 5. Análisis de los lenguajes de programación empleados en el campo de la Inteligencia Artificial. 17

Tabla 6. Comparación de Plataformas de desarrollo y gestión de datos 18

Tabla 7. Ventajas y Desventajas de los métodos usados en Ecuador. 20

Tabla 8. Comparación de las soluciones de visión artificial más utilizadas en la agricultura de precisión. 21

Tabla 9. Sistema físico de Visión Artificial. 21

Tabla 10. Visión artificial y sus aplicaciones. 24

Tabla 11. Comparación de los drones utilizados para el conteo de frutos y la estimación de producción. 29

Tabla 12. Estrategias de control aplicadas a cuadrotores. 31

Tabla 13. Factores clave en la planificación del vuelo. 32

Tabla 14. Comparación de características de aplicaciones para la planificación de trayectorias. 37

Tabla 15. Comparación entre YOLOv8 y VGG16 aplicadas a la clasificación y análisis de cultivos. 45

Tabla 16. Comparación de microcomputadores. 49

Tabla 17. Componentes importantes de una SGBD. 50

Tabla 18. Comparación de sistema de gestión de base de datos. 53

Tabla 19. Características del hardware seleccionado. 56

Tabla 20. Características del software empleado. 56

Tabla 21. Criterios de recolección de información. 58

Tabla 22. Parámetros de configuración del modelo. 95

Tabla 23. Resultados del Entrenamiento. 96

Tabla 24. Métricas del Entrenamiento 96

Tabla 25. Comparación de modelos. 105

Tabla 26. Resultados de la primera prueba. 116

Tabla 27. Pruebas de funcionamiento. 118

Tabla 28. Tabla de estimación de producción por fila (kg/fila) 119

Tabla 29. Tabla de estimación para las 21 filas 119

Tabla 30. Pruebas de funcionamiento para la predicción de cosecha. 120

Tabla 31. Eficacia total del sistema. 122

Tabla 32. Requisitos mínimos de hardware. 123

Tabla 33. Requerimientos mínimos de software 124

Tabla 34. Costos computacionales. 125

Tabla 35. Presupuesto de implementación del sistema. 126

ÍNDICE DE FIGURAS

Figura 1. Provincias con la mayor producción de mora en Ecuador. 9

Figura 2. Mora Castilla. 10

Figura 3. Estados de madurez de la mora castilla. 10

Figura 4. Mora Colombiana. 11

Figura 5. Mora Brazos. 11

Figura 6. Mora INIAP Andimora 12

Figura 7. Boletín Informacional del Cultivo de Mora 2023. 16

Figura 8. Subsistemas físicos de un equipo de visión artificial. 22

Figura 9. Organización matricial uniforme de una imagen digital. 23

Figura 10. Etapas de una aplicación de visión artificial. 24

Figura 11. Clasificación de los UAVs. 25

Figura 12. Ala Fija. 26

Figura 13. Helicóptero. 27

Figura 14. Multirrotor. 27

Figura 15. Diagrama de un sistema de control en lazo cerrado. 29

Figura 16. Control punto a punto. 30

Figura 17. Seguimiento de trayectorias. 30

Figura 18. Evasión de obstáculos. 30

Figura 19. Delimitación del área. 33

Figura 20. Selección del dron. 33

Figura 21. Waypoints con dron. 34

Figura 22. Control de dron manualmente. 35

Figura 23. Aplicación Dronelink..... 35

Figura 24. Aplicación Rainbow for DJI Drones. 36

Figura 25. Aplicación Drone Harmony. 36

Figura 26. Regresión Lineal Simple utiliza una única variable independiente. 38

Figura 27. Comportamiento de los patrones en una regresión polinómica. 39

Figura 28. Correlación entre datos extensos. 39

Figura 29. Árboles de Decisión..... 40

Figura 30. Bosques aleatorios. 41

Figura 31. Arquitectura de Redes Neuronales Convolucionales 42

Figura 32. Redes Neuronales Recurrentes (RNN). 42

Figura 33. NVIDIA Jetson Nano. 47

Figura 34. Raspberry Pi 5..... 48

Figura 35. Intel Neural Compute Stick. 48

Figura 36. Base de datos SQLite. 52

Figura 37. Base de datos PostgreSQL. 52

Figura 38. Administrador de base de datos MongoDB. 53

Figura 39. Ubicación del cultivo de mora. 60

Figura 40. Cultivo de mora desde tomada con el dron. 60

Figura 41. Etapas del Esquema general del sistema. 64

Figura 42. Esquema Funcional del sistema. 65

Figura 43. Etapa 1 adquisición de datos..... 66

Figura 44. Etapa 2 entrenamiento del modelo. 66

Figura 45. Capas de Algoritmo 67

Figura 46. Etapa 3 de procesamiento de los datos y control de vuelo. 68

Figura 47. Etapa 4 Visualización y almacenamiento de datos..... 69

Figura 48. Diagrama de flujo general del algoritmo del sistema. 70

Figura 49. Escenario del sistema en el cultivo de mora. 71

Figura 50. Diagrama de flujo del control de vuelo autónomo. 72

Figura 51. Conexión del dron con la aplicación. 74

Figura 52. Interfaz de Usuario de la aplicación Rainbow for DJI Drones. 74

Figura 53. Modos de vuelo del dron. 76

Figura 54. Parámetros importantes en modo Way point. 76

Figura 55. Selección del primer icono para los puntos de referencia. 77

Figura 56. Puntos de referencia establecidos para copar toda el área del cultivo de mora..... 78

Figura 57. Altura establecida para el dron. 78

Figura 58. Velocidad establecida para el dron. 79

Figura 59. Diagrama de flujo para la adquisición de video. 79

Figura 60. Diagrama de flujo para el procesamiento de datos..... 83

Figura 61. Imagen original RGB frente a imagen convertida a HSV 86

Figura 62. Anotación de datos utilizando Labelme. 90

Figura 63. Arquitectura del dataset 91

Figura 64. Archivo dataset.yaml 92

Figura 65. Cargando el modelo Yolov8m 93

Figura 66. Entrenando el modelo por 50 Épocas 93

Figura 67. Librerías para el modelo VGG16. 94

Figura 68. Parámetros del entrenamiento. 94

Figura 69. Instanciación de directorios de los datos. 95

Figura 70. Codificación para ajustar los datos a las necesidades del modelo. 95

Figura 71. Métricas de rendimiento del modelo entrenado (Detección de moras). ... 98

Figura 72. Matriz de Confusión. 99

Figura 73. Matriz de Correlación. 101

Figura 74.Predicción por lote modelo detección de moras 102

Figura 75. Curva de Precisión y Recuperación 102

Figura 76. Curva de Precisión 103

Figura 77. Curva de exactitud modelo VGG16. 104

Figura 78. Curva de pérdidas del modelo VGG16. 105

Figura 79. Base de datos de las moras en MongoDB. 106

Figura 80. Diagrama entidad-relación del sistema de base de datos. 106

Figura 81. Interfaz de autenticación de usuario. 108

Figura 82. Botones principales de la interfaz gráfica. 108

Figura 83. Detección y conteo de la mora..... 109

Figura 84. Tablas de detección de moras. 109

Figura 85. Cálculo de costo y peso de moras. 110

Figura 86. Graficas de peso y costo de las moras. 110

Figura 87. Interfaz de predicción de la cantidad de la futura cosecha. 111

Figura 88. Esquema prototipo del sistema. 112

Figura 89. Interfaz Jetson Nano. 113

Figura 90. Configuración de OpenCV en la Jetson Nano. 114

Figura 91. Inicio de la cámara en la primera prueba de funcionamiento del sistema. 115

Figura 92. Detección en tiempo real. 116

Figura 93. Cálculo y resultado de la estimación de la producción..... 116

Figura 94. Predicción meteorológica del del clima..... 117

Figura 95. Simulación de Montecarlo en la interfaz del sistema. 118

ÍNDICE DE ANEXOS

Anexo A. UAV DJI Mini2 135

Anexo B. NVIDIA Jetson Nano..... 136

Anexo C. Interfaz Web. 136

Anexo D. Hardware Adquirido. 137

Anexo E. Cultivo de mora..... 138

Anexo F.Código del sistema de estimación. 140

Anexo G. Abreviaturas de palabras de la tesis. 148

RESUMEN EJECUTIVO

El cultivo de mora es uno de los sectores importantes de la economía nacional debido a su alta demanda tanto a nivel local como internacional. Sin embargo, los agricultores enfrentan desafíos importantes al intentar predecir la producción con exactitud. Los métodos convencionales, que dependen en gran medida de procedimientos manuales, suelen ser imprecisos y propensos a errores humanos, lo que puede comprometer la planificación y generar pérdidas económicas. EL sistema desarrollado en esta investigación se compone de cuatro etapas fundamentales: adquisición, procesamiento, entrenamiento y análisis. Para la recolección de datos, se utilizó un vehículo aéreo no tripulado (VANT) DJI Mini 2, capaz de capturar imágenes en tiempo real de los cultivos. Durante el procesamiento, dichas imágenes fueron analizadas mediante técnicas de visión por computadora, empleando herramientas como OpenCV y el modelo de detección YOLOv8m. El modelo fue entrenado con un conjunto de datos específico que incluía fotografías de moras en distintas etapas de madurez: verde, roja y morada. Además, se integró una simulación basada en Montecarlo para proyectar posibles pérdidas ocasionadas por factores como plagas, condiciones climáticas adversas y sequías. Los resultados se presentaron en una interfaz gráfica diseñada para facilitar la visualización y el análisis, ayudando a los productores a optimizar su toma de decisiones. El sistema demostró una confiabilidad del 80,55% en la identificación y conteo de moras, con datos capturados directamente en campo. Asimismo, las simulaciones ofrecieron una evaluación detallada de posibles escenarios adversos, permitiendo estimaciones más completas y ajustadas a la realidad.

Palabras claves: Visión artificial, estimación de producción, vehículo aéreo no tripulado (VANT), Modelo YOLOv8m, agricultura de precisión.

ABSTRACT

Blackberry cultivation is a significant sector of the national economy due to its high demand at both local and international levels. However, growers face considerable challenges in attempting to accurately predict production. Conventional methods, which rely heavily on manual procedures, are often inaccurate and prone to human error, potentially compromising planning and resulting in economic losses. The system developed in this research is composed of four fundamental stages: acquisition, processing, training, and analysis. For data collection, a DJI Mini 2 unmanned aerial vehicle (UAV) was utilized, capable of capturing real-time images of the crops. During processing, these images were analyzed using computer vision techniques, employing tools such as OpenCV and the YOLOv8m detection model. The model was trained with a specific dataset that included photographs of blackberries at different stages of maturity: green, red, and purple. To project potential losses due to factors such as pests, adverse weather, and drought, a Monte Carlo-based simulation was integrated. The results were presented in a graphical interface designed to facilitate visualization and analysis, assisting growers in optimizing their decision-making. The system exhibited 80.55% reliability in blackberry identification and counting, with data captured directly in the field. Furthermore, the simulations yielded a comprehensive evaluation of potential adverse scenarios, enabling more precise and realistic estimates.

Keywords: Artificial vision, yield estimation, unmanned aerial vehicle (UAV), YOLOv8m model, precision agriculture.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA DE ESTIMACIÓN DE PRODUCCIÓN DE CULTIVO DE MORA EMPLEANDO VISIÓN ARTIFICIAL Y MACHINE LEARNING

1.1.1 Planteamiento del problema

La producción de mora es una actividad agrícola de gran relevancia económica en diversas provincias del Ecuador. Sin embargo, los métodos tradicionales de estimación de producción suelen ser manuales, laboriosos y sujetos a errores humanos, lo cual puede derivar en una planificación ineficiente y pérdidas económicas para los agricultores. La precisión en la estimación de la producción es crucial para la toma de decisiones en todas las etapas de la cadena de suministro, desde la siembra hasta la comercialización.

En Latinoamérica, Colombia se destaca como el país con mayor producción de mora. Al comparar la producción de mora entre Colombia y Ecuador en los años 2017, 2018 y 2019, se observa que Colombia tuvo una producción significativamente mayor. En 2017, Colombia produjo 123,175 toneladas de mora, en 2018 alcanzó aproximadamente 129,977 toneladas, y en 2019 la producción aumentó a 143,145 toneladas, siendo este el año con mayor producción en Colombia. En contraste, Ecuador, donde la producción de mora se concentra en solo 8 provincias de la región Sierra, alcanzó su mayor producción en 2017 con 17,909 toneladas. Sin embargo, en 2018, la producción en Ecuador disminuyó un 4% [1]. Ecuador está por debajo de Colombia rendimiento por hectárea de producción de mora. En 2021, el rendimiento de mora en Colombia fue de 11,16 toneladas por hectárea, mientras que en Ecuador fue de 4,69 toneladas por hectárea, menos de la mitad del rendimiento colombiano. Esta diferencia se debe a la mayor cantidad de áreas dedicadas a la producción de mora en Colombia en comparación con Ecuador. [2]

En el año 2022, la producción nacional de mora en Ecuador alcanzó las 13,438 toneladas. Para Ecuador las provincias importantes en producción de mora siguen siendo las misma que en anteriores años tales como Imbabura, Pichincha, Cotopaxi, Tungurahua, Chimborazo y Bolívar. Estas provincias destacan por su extensa producción al cultivo de mora, contribuyendo significativamente a la producción total del país. Además, existen otras provincias no especificadas por el Ministerio de Agricultura y Ganadería cuya producción de mora es considerablemente menor en comparación con las mencionadas anteriormente. Bolívar, reconocida por su terreno propicio y clima favorable, continúa siendo uno de los pilares de la producción de mora en Ecuador. Tungurahua y Cotopaxi, con sus condiciones geográficas privilegiadas, también juegan un papel importante en el sector. Chimborazo e Imbabura, conocidas por su diversidad agrícola, contribuyen con volúmenes significativos de mora cada año. Pichincha, con su proximidad a mercados urbanos clave, complementa la cadena de suministro con su producción de alta calidad. [2]

En el año 2022, Tungurahua se destacó como la segunda mejor provincia en producción de mora en Ecuador, alcanzando un total de 6,373 toneladas. Esta provincia no solo se posicionó como una de las principales productoras a nivel nacional, sino que también demostró el mejor rendimiento por hectárea con 7.24 toneladas, superando a todas las demás provincias del país en este aspecto. [2]

La supervisión manual de cultivos en la agricultura tradicional presenta significativas limitaciones debido a la falta de datos detallados y en tiempo real sobre el estado de las plantas. Esta metodología de recolección de información, que depende en gran medida de la observación humana y las visitas periódicas al campo, no puede capturar cambios dinámicos y rápidos en las condiciones del cultivo. Además, las observaciones entre diferentes personas tienen a tener errores que pueden distorsionar la percepción de la salud del cultivo. La supervisión manual también resulta poco práctica y costosa a medida que el tamaño de las producciones agrícolas aumenta se requiere una cantidad considerable de tiempo y mano de obra para realizar inspecciones exhaustivas y frecuentes.

Actualmente, los métodos para predecir la producción de mora en Ecuador carecen de precisión debido a su dependencia de diversas técnicas como modelos estadísticos, datos históricos y observaciones de campo. Estas metodologías, aunque son fundamentales, son propensas a errores. Por un lado, las estimaciones excesivas pueden llevar a una sobreestimación de la producción de mora, lo cual conlleva una planificación excesiva de recursos como mano de obra, equipos y espacio de almacenamiento, potencialmente resultando en desperdicio si la producción real no alcanza las expectativas. Por otro lado, las estimaciones deficientes pueden subestimar la producción, lo que puede conducir a una escasez de recursos planificados y, en última instancia, a una cosecha insuficiente.

La optimización y automatización de la gestión del cultivo mediante tecnologías avanzadas, como la visión artificial y el aprendizaje automático, ofrecen un potencial significativo para mejorar la precisión en la estimación de la producción y la eficiencia en la gestión de los recursos agrícolas. La visión artificial permite el monitoreo continuo y detallado de los cultivos a través de imágenes y sensores, proporcionando información en tiempo real sobre el estado de las plantas.

1.2 Antecedentes investigativos

Se han recopilado varios trabajos investigativos de varios repositorios que pueden aportar al trabajo de investigación en cuanto a sistemas de estimación como tecnologías aplicadas se detallan a continuación:

En el año 2019, O. E. Apolo-Apolo, J. Martínez-Guanter, G. Egea y M. Pérez-Ruiz presentaron su investigación titulada “Estimación de producción en cítricos usando técnicas de aprendizaje automático” en el X Congreso Ibérico de Agroingeniería. Este trabajo abordó la necesidad crucial de estimar con precisión la cosecha de cítricos, información vital para la planificación logística, el almacenaje y el abastecimiento de mercados. Tradicionalmente, estas estimaciones se realizan mediante inspección visual y datos históricos, métodos que pueden estar sujetos a errores humanos. La investigación propuso el uso de redes neuronales convolucionales (CNN) basadas en Deep Learning (DL) para el reconocimiento y conteo de frutos, con el objetivo de crear

un modelo que estimara la producción de cítricos a partir de imágenes capturadas por drones. Los modelos entrenados se probaron en imágenes de 20 árboles de cítricos seleccionados al azar, obteniendo un error medio absoluto de entre 4-6%, en comparación con el 8-11% del error de estimación realizado por técnicos especialistas. Estos resultados evidenciaron el potencial significativo de las CNN para mejorar la precisión en la estimación del rendimiento de cítricos. No obstante, el estudio también resaltó las limitaciones actuales, como la capacidad de cómputo y las dificultades para detectar frutos en condiciones de oclusión y sombra. A pesar de estas limitaciones, los investigadores expresaron optimismo sobre la posibilidad de desarrollar modelos más precisos y robustos para su aplicación en explotaciones comerciales de cítricos. [3]

En el año 2020, el MsC. Carlos Alejandro Ramírez Gómez, presentó su investigación titulada “Aplicación del Machine Learning en Agricultura de Precisión” en la Revista Cintex. Este artículo propone un modelo de Machine Learning para predecir el estado de la cosecha basado en información sobre el consumo de pesticidas y otras variables del cultivo. La metodología seguida incluye cuatro pasos principales: preprocesamiento y análisis de la información, separación de los datos en conjuntos de entrenamiento, prueba y validación, selección de los modelos, y evaluación de los hiperparámetros utilizando una métrica específica. Se propusieron y evaluaron cinco modelos de clasificación, utilizando el "accuracy score" como métrica de evaluación. Los resultados indicaron que los modelos KNN y Decision Tree eran los más prometedores. A través de un análisis de sensibilidad, se determinaron los hiperparámetros óptimos para estos modelos: 200 vecinos para KNN y una profundidad de 5 para el árbol de decisión. Comparando ambos modelos, se encontró que el árbol de decisión presentaba un mejor desempeño con menor coste computacional. Finalmente, se desarrolló un modelo de ensamble tipo Boosting basado en árboles de decisión, logrando una mejor aproximación y seleccionándose como el modelo definitivo para la predicción del estado de la cosecha. Este estudio demuestra el potencial de los algoritmos de Machine Learning para mejorar la precisión y eficiencia en la agricultura de precisión. [4]

En el año 2021 el Sr. Daniel Medina Tobón presentó su investigación titulada “Conteo de flores y frutos para el monitoreo del cultivo de aguacate Hass por medio de imágenes utilizando Machine Learning”. Este estudio desarrolló una aplicación capaz

de realizar un conteo automático de flores y frutos en árboles de aguacate Hass a partir de imágenes de baja resolución obtenidas con una cámara de bajo costo. Se utilizaron arquitecturas de redes neuronales como YOLOv5 y Faster RCNN-ResNet50 para la detección de objetos en las imágenes. Aunque la precisión del modelo Faster RCNN fue baja y se descartó, el modelo YOLOv5 demostró ser prometedor, logrando una precisión del 80% en el conteo de aguacates y del 39% en el conteo de flores. Este sistema proporciona información valiosa para el seguimiento de cada árbol, permitiendo acciones tempranas para corregir problemas de crecimiento, experimentar con nuevos tratamientos y generar un historial de datos útil para predecir el rendimiento. El desarrollo se realizó con recursos de bajo costo, utilizando una cámara accesible y Google Colab para el entrenamiento de los modelos, mostrando la viabilidad de implementar soluciones económicas y eficaces para el monitoreo de cultivos. Este trabajo, parte de una tesis doctoral, representa un componente inicial en el desarrollo de un sistema integrado para el monitoreo de cultivos, cumpliendo con su objetivo de ofrecer una herramienta útil y accesible para agricultores. [5]

En la investigación llevada a cabo por el Ing. Andrés Alejandro Garcés Cadena en la Universidad Técnica de Ambato en 2022, su investigación titulada "Sistema cuantificador de calidad de cultivo de manzana para monitoreo de la producción utilizando algoritmos de Aprendizaje Profundo con Visión Artificial y Segmentación de Instancias" tuvo como objetivo proporcionar una herramienta para mejorar la gestión de los cultivos de manzana mediante el uso de algoritmos de Deep Learning y Visión por Computador. El sistema desarrollado realizaba dos análisis principales: la detección del tipo de manzana y la cuantificación de su calidad mediante un método no invasivo. Para la detección, se utilizó el modelo SSD-Mobilenet, y para la segmentación basada en la calidad, se empleó una red convolucional FCNResNet-18, ambas redes fueron reentrenadas con bases de datos personalizadas específicas para el proyecto. La evaluación del sistema se llevó a cabo utilizando métricas como matrices de confusión y solapamiento de objetos (IoU). Los resultados mostraron que el modelo SSD-Mobilenet ofrecía una respuesta más rápida en la detección de objetos en comparación con otros modelos como deeplab-VGG e Inception V2. Este trabajo destacó el potencial de las técnicas de Deep Learning y Visión Artificial para mejorar la precisión y eficiencia en la gestión de cultivos agrícolas. [6]

El Dr. Sabab Ali Shah presentó en 2023 su trabajo de investigación titulado "Application of Drone Surveillance for Advance Agriculture Monitoring by Android Application Using Convolution Neural Network" en el cual demostró la viabilidad de la visión artificial para monitorear el crecimiento y el estado de los cultivos. En su investigación, desarrolló un sistema que utiliza drones equipados con cámaras y una aplicación Android, empleando redes neuronales convolucionales (CNN) para analizar las imágenes capturadas. El sistema permitió una evaluación precisa y eficiente de la salud de los cultivos, identificando problemas como enfermedades o falta de nutrientes de manera temprana. Además, en el mismo estudio se exploró el uso de algoritmos de machine learning para predecir rendimientos agrícolas con alta precisión, basándose en los datos recogidos por los drones. Los resultados obtenidos destacaron la capacidad de estas tecnologías para proporcionar información crítica a los agricultores, optimizando así las prácticas agrícolas y contribuyendo a un mejor manejo de los recursos agrícolas. [4]

Este conjunto de investigaciones aporta de forma significativa al proyecto ya que incluye factores importantes en cuanto a la viabilidad de la visión artificial, machine learning y pruebas de prototipos para la estimación de producción en diferentes frutos y plantas, otro aspecto importante que proveen las investigaciones son una serie de mediciones y parámetros a considerar al momento del diseño del prototipo junto con la tecnología que se utiliza para realizar la adquisición y procesamiento de las imágenes.

1.3 Fundamentación teórica

1.3.1 Agricultura de Precisión

La agricultura de precisión es un tipo de sistema de gestión agrícola que optimiza el uso de recursos naturales y económicos mediante el análisis detallado de la información recogida en tiempo real de los cultivos y terreno. Hay varias herramientas que contribuyen a optimizar los diversos procesos agronómicos, permitiendo mejorar

la toma de decisiones y el uso eficiente de los recursos que son importantes para las actividades agrícolas, tales como sensores remotos, satélites, GPS y drones. [5]

Los beneficios de aplicar agricultura de precisión es la mejora en el rendimiento de los cultivos, reducción de costos para el uso de los insumos agrícolas y la disminución del impacto ambiental. Al monitorear de manera continua y en tiempo real los cultivos, los agricultores pueden ayudarse de manera significativa para las necesidades del campo, mejorando la eficiencia en los procesos agrícolas. [5]

1.3.2 Tecnologías involucradas en la agricultura de precisión

En la Tabla 1 se encuentran las tecnologías fundamentales para la agricultura de precisión se encuentran:

Tabla 1. Tecnologías usadas comúnmente en la agricultura de precisión. [5]

Sistemas de Posicionamiento Global (GPS)	Son empleados en la geolocalización precisa de maquinaria agraria y en el monitoreo de operaciones en amplias áreas de terreno. Esta herramienta facilita el monitoreo de los cultivos a lo largo del tiempo, mejora la precisión en la siembra y en la aplicación de fertilizantes y otros insumos.
Sensores remotos	Los drones y satélites están equipados con sensores, son importantes para la recolección de imágenes y datos que pueden ser procesados para generar mapas de vigor de los cultivos, deficiencias nutricionales y presencia de plagas o enfermedades. Por lo que los drones pueden monitorear minuciosamente áreas agrícolas capturando imágenes que después son analizados por medio de un software especializado y brindan datos sobre las condiciones del cultivo.
Sistemas de Información Geográfica (SIG)	Estos sistemas permiten un análisis espacial detallado de los cultivos y el suelo. Los agricultores pueden usar estas herramientas para identificar áreas con problemas de fertilidad, plagas o exceso de humedad, lo que facilita la implementación de medidas correctivas específicas.

a. Aplicación de drones en la agricultura de precisión

Debido a su capacidad para realizar vuelos autónomos y recopilar datos precisos sobre grandes áreas en poco tiempo, los drones son esenciales para la agricultura de precisión. Los drones están equipados con cámaras y sensores que pueden capturar imágenes y transformarlas en mapas que muestran detalles sobre el estado de los cultivos. Estos dispositivos permiten a los agricultores hacer un seguimiento cercano

del desarrollo del cultivo sin tener que estar físicamente en el campo, lo que ahorra tiempo y reduce los costos. [5]

b. Ventajas de la agricultura de precisión

Las principales ventajas de la agricultura de precisión son:

Mejor uso de los recursos: Permite utilizar el agua, los fertilizantes y los pesticidas de manera eficiente, aplicándolos solo cuando es necesario. [5]

Reducción de costos: Al utilizar los insumos de manera más eficiente, se reducen los costos de producción. [5]

Incremento en la productividad: Las intervenciones mejoran el rendimiento y la calidad de los cultivos al adaptarlas a las necesidades específicas de cada parcela. [5]

Monitoreo remoto: Los agricultores pueden monitorear sus cultivos a distancia a través de drones y satélites, lo que mejora la eficiencia operativa. [5]

1.3.3 Cultivo de Mora

La mora es una planta que se encuentra esencialmente en los Andes ecuatorianos, adaptada a climas fríos y templados. Comúnmente crecen en diversos entornos como quebradas, matorrales, montes, y a lo largo de caminos y carretera. [6]

En Ecuador, la producción de mora se concentra principalmente en la zona de la Sierra, siendo 9 de las 24 provincias del país las más destacadas en producción de mora. En la Tabla 2 se muestra las provincias de Ecuador y su gran aporte en la producción de mora. [7]

Tabla 2. Provincias con la mayor producción de mora en Ecuador. [7]

Provincia	Superficie (ha)	Producción (t)	Rendimiento (t/ha)
1. Tungurahua	868	7.098	8.18
2. Cotopaxi	551	5.066	9.19
3. Bolívar	1.216	4.283	3.52
4. Pichincha	88	1.134	12.94
5. Zamora Chinchipe	38	125	3.27
6. Imbabura	60	123	2.03
7. Otros	60	232	3.84
Nacional	2.882	18.061	6.27

En la Figura 1 se muestra las provincias con la mayor producción de mora en Ecuador. [7]

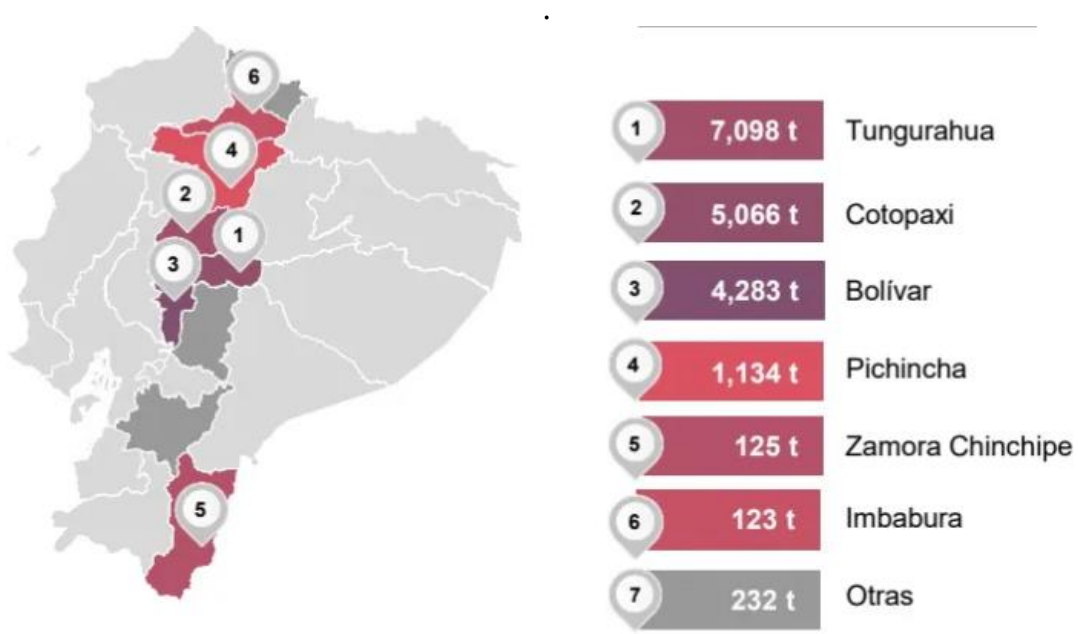

Figura 1. Provincias con la mayor producción de mora en Ecuador. [7]

En Ecuador, hay varias especies de moras que se producen las cuales son: la mora Castilla (Rubus glaucus), colombiana (Rubus glaucus Benth), Brazos (Rubus lanciniatus) y INIAP Andimora 2013. [6]

a. Mora de Castilla (Rubus glaucus)

Este tipo de mora se cultiva en la parte de región Sierra del país, se caracteriza por que sus ramas largas de aproximadamente 3.22m, con presencia de espinas, su fruto pesa alrededor de 4.5g, además de que su ciclo productivo es tardío con 186 días de brotación para su cosecha. [8]

En la Figura 2 se muestra el fruto que es un grupo de pequeñas drupas de color rojo púrpura o morado brillante, con una forma cónica ovalada y una punta redondeada. Miden entre 3 y 4 cm de largo y 3 a 4 cm de diámetro. [6]

Cuando el fruto no está totalmente maduro, tiene un sabor agridulce, mientras que cuando está completamente maduro, tiene un sabor dulce, de color negro morado oscuro brillante. [6]

Figura 2. Mora Castilla. [6]

La mora Castilla tiene varias etapas de maduración que se mencionan a continuación: [9]

- ✓ Color 0: El fruto en su totalidad es de color verde con pocas drupas marrones.
- ✓ Color 1: El fruto es de color verde claro con algunas semillas rosadas o rojas.
- ✓ Color 2: El fruto tiene una tonalidad roja con algunas drupas amarillas.
- ✓ Color 3: El fruto tiene un color rojo intenso con rasgos de líneas moradas.
- ✓ Color 4: El fruto es de color morado oscuro y casi en su totalidad negro.

En la Figura 3 se muestra los estados de madurez de la mora de Castilla:

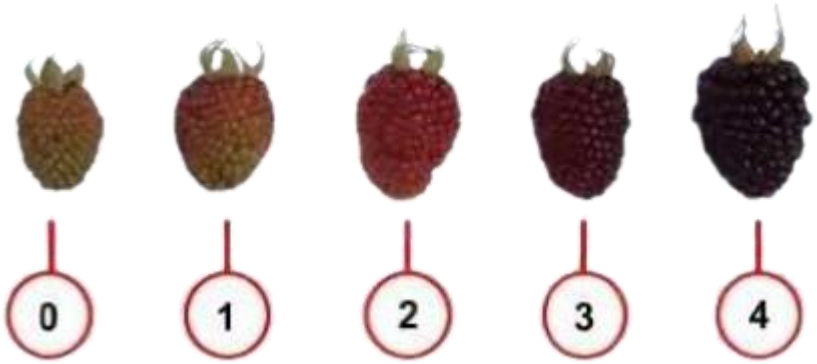

Figura 3. Estados de madurez de la mora castilla. [9]

b. Mora Colombiana (Rubus glaucus Benth)

Este tipo de mora se caracteriza por tener espinas además de ser más precoz que la mora Castilla con 161 días de brotación para su cosecha. En la Figura 4 se muestra a

los frutos de la mora colombiana, sus frutos son pequeños de aproximadamente 3.96g, además estas son sensibles a heladas y enfermedades. [8]

Figura 4. Mora Colombiana. [6]

c. Mora Brazos (Rubus lanciniatus)

Este tipo de mora es originaria de Texas, se destaca por su fuerza y la presencia de grandes espinas, produciendo frutos de mayor tamaño de 3,0 a 5,0 gramos. En la Figura 5 el fruto tiene una consistencia firme y un sabor agridulce, y se compone de pequeñas drupas adheridas a un receptáculo carnoso y desarrollado. Su color cambia de rojo a negro brillante a medida que madura. Además, se adapta mejor a climas cálidos y tiene un ciclo productivo de alrededor de 164 días. [6] [8]

Figura 5. Mora Brazos. [8]

d. Mora INIAP Andimora

Desarrollada por el INIAP, es una variedad sin espinas que facilita la cosecha. Tiene el ciclo de producción de 171 días. Es adecuada para la agroindustria por su firmeza y buena calidad de fruta. En la Figura 6 se muestra el aspecto físico de la INIAP Andimora. [6]

Figura 6. Mora INIAP Andimora [8]

e. Tabla comparativa de descriptores morfológicos

En la Tabla 3 se establecen descriptores cuantitativos y cualitativos morfológicos para diferenciar los 4 tipos de mora que tienen un promedio de ($p<0.05$). [8]

Tabla 3. Descriptores Cuantitativos y cualitativos morfológicos. [8]

Descriptor	**Castilla**	**Andimora**	**Colombiana**	**Brazos**
Longitud de tallo (m)	3.22	3.38	2.18	1.66
Diámetro de tallo (mm)	8.50	8.33	7.47	6.17
Tamaño de espinas en rama productivas (mm)	3.40	0.00	0.00	8.29
Longitud de fruto (mm)	22.54	26.58	22.34	25.31
Diámetro de fruto (mm)	20.15	20.55	19.61	22.03
Peso por fruto (g)	4.50	4.52	3.96	5.85
Tipo de hoja	Trifoliada	Trifoliada	Trifoliada	Palmada
Forma del pétalo	Lanceolado	Lanceolado	Lanceolado	Redondeado
Forma del fruto	Oval estrecha	Cónica alargada	Oval estrecha	Oblonga

1.3.4 Métodos para la producción de mora

Para el cultivo de mora es necesario varios factores tales como: las condiciones ambientales, el establecimiento de donde se va a realizar el cultivo de mora y los sistemas de propagación de mora. [10]

a. Condiciones Ambientales

El cultivo de mora requiere de condiciones ambientales específicas para alcanzar su máximo potencial de producción y asegurar la calidad del cultivo de mora. Estas condiciones involucran factores clave como la altitud, la humedad, la temperatura, las

precipitaciones, la luz solar y la calidad del suelo. En la siguiente Tabla 4 se muestran las características de las condiciones ambientales. [10]

Tabla 4. Características de las condiciones ambientales para el cultivo de mora. [10]

Características	**Descripción**
Altitud	Las condiciones ambientales son necesarias ya que para que la planta de mora alcance su máximo nivel de producción debe estar entre 1800 s 2400 metros de altura sobre el nivel del mar. Después de los 2400 metros el rendimiento de la planta disminuye por lo que la calidad del tamaño de los frutos es menor.
Humedad	La humedad requerida se establece entre 70% y 80%.
Temperatura	Oscila entre los 11 °C a los 18°C.
Precipitaciones de lluvia	Las precipitaciones de lluvia entre 1500 a 2500 milímetros son las más aptas
Luminosidad	Los cultivos requieres de mínimo 1200 1600 horas de luz solar al año.

b. Establecimiento del cultivo

El suelo es una de las principales características para el cultivo de mora por lo que el suelo ideal debe de ser de textura franca, rico en materia orgánica, la cual debe tener humedad. [10]

La selección del lote para el cultivo de mora los suelos francos son importantes, además deben tener una buena aireación y que no se encorchen, esto para que las raíces puedan penetrar con facilidad en el suelo. [10]

En la preparación del terreno se realiza arado con cincel y dos pases con discos de manera técnica y no muy generalizada para otros tipos de cultivos. [10]

Al ver establecido el suelo, se traza y se prepara el hoyo, hay dos tipos de trazos uno de ellos es el trazo en cuadro que se realiza cuando el terreno es plano o tiene pendientes menores de 5%. Se define la dirección de los surcos, tendiendo una cabuya por donde se va a establecer el primer surco, con la ayuda de una estaca se establece la distancia de 2 metros que debe existir entre plantas del mismo surco, y para la conformación de los demás surcos se recomienda una distancia de 2.5 metros teniendo en cuenta el primer surco para así formar una escuadra entre surcos. El trazo a través de la pendiente se lo realiza en pendientes medias pronunciadas las distancias entre plantas es de 2 metros y entre surcos de 2.5 metros al igual que en el trazo en cuadrado, se inicia el trazo de los surcos desde la parte más ancha del terreno, mediante dos varas

de 2 metros y 2.5 metros se forma una T la cual se coloca en la cabuya, teniendo en cuenta que la vara más corta se establece el primer surco y el segundo surco se establece la distancia con la vara más larga, así de manera simultánea se van alineando las varas hasta establecer todos los surcos en el terreno. [10]

El hoyado del suelo se debe agregar un 1 kilogramo de materia orgánica, las medidas recomendadas del hoyo son de 40x40x40 cm. Las densidades de las siembras van a variar por las condiciones del terreno, la fertilidad y por la humedad ambiental, si el suelo es muy fértil las distancias de las simbras pueden ser hasta 2.5 metros. Las distancias adecuadas entre las plantas de mora permiten la aireación, y con eso se puede manejar de mejor manera las enfermedades y las plagas. [10]

c. Sistemas de propagación de mora

Existen 2 tipos de sistemas de propagación de mora, por semilla o por material vegetal de los tallos y raíces. [10]

La primera es la propagación por acodo que es una de las más conocidas la cual consiste en introducir la punta de una rama de mora en la tierra sin dejar ninguna hoja en la rama a plantar. [10]

El otro sistema de propagación de mora es la propagación por cepa consiste en sacar las cepas de las plantas mora que estén más sanas y fuertes en la producción de mora. [10]

1.3.5 Factores que afectan a la producción de mora

A continuación, se detallan las principales causas que afectan a la superficie de cultivo de mora, como: plagas, sequías y otras causas en general.

a. Plagas

Las plagas ocupan un 24% en afectación de la superficie de cultivo de mora, siendo una afectación directa de organismos dañinos que atacan a las plantas y sus frutos, reduciendo el rendimiento, y hasta en casos extremadamente graves afectan a todo el cultivo. [7]

b. Sequia

La sequía tiene un 23% en afectación a la superficie de cultivo de mora, por lo que una gestión deficiente de riego o por falta de agua prolongado afecta al desarrollo de las plantas reduciendo el crecimiento vegetativo y el tamaño de los frutos, por lo que existiría una disminución considerable de producción o en casos extremos puede llevar a la muerte de los cultivos de mora. [7]

c. Otras causas

Tiene un 53% de influencia en la afectación de la superficie de cultivo de mora, de este conjunto de otras causas influyen factores como los económicos, heladas y enfermedades hacia el cultivo de mora. [7]

- ***Heladas (20%)***

El cultivo de mora por estar centrado en la región Sierra tiende a tener bajas temperaturas en las noches y esto daña los tejidos de las plantas, especialmente en la etapa de floración y en el desarrollo tempano de los frutos. [7]

- ***Enfermedades (15%)***

Las principales enfermedades que afectan a los cultivos de mora son los hongos y patógenos que se dan en las raíces y en la parte superior de la planta de mora como. [7]

- ***Causas económicas (18%)***

Los productores de mora en Ecuador tienden a tener bajos recursos económicos para comprar insumos que son muy costosos y afectan principalmente a la calidad y perdida de producción de la mora. [7]

Según el Ministerio de agricultura del Ecuador existe unas pérdidas de superficie de 82 ha a nivel nacional por las causas expuestas anteriormente que son importante por los relieves y clima del Ecuador, en la Figura 7 se observa que plagas y enfermedades, sequía y otras causas provocan este problema. [7]

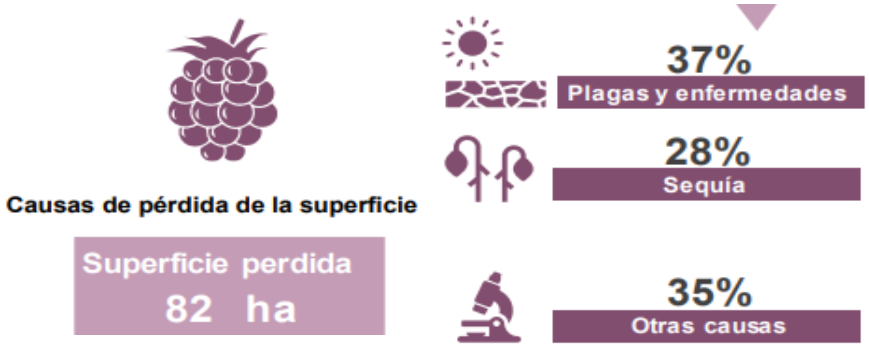

Figura 7. Boletín Informacional del Cultivo de Mora 2023. [7]

1.3.6 Inteligencia Artificial en la Agricultura

La inteligencia artificial (IA) forma parte de la informática y tiene como objetivo desarrollar sistemas que sean capaces de realizar funciones que demandan inteligencia humana, como decidir, reconocer patrones y aprender de la información. Dentro del campo agrícola, la IA ha tomado relevancia por su efectividad en optimizar procesos de producción, mejorar la gestión de cultivos y reducir su impacto sobre el medio ambiente. [11]

El uso de la IA en la agricultura se centra en el manejo de gran cantidad de datos producido por tecnologías como satélites, drones y sensores, lo que permite identificar patrones y realizar decisiones informadas al instante. Toda esta información es guardada en un sistema de gestión de Base de Datos. Estos sistemas pueden prever el rendimiento agrícola, reconocer ciertas enfermedades o plagas y optimizar recursos como fertilizantes, agua, entre otros. [11]

a. Lenguajes de programación utilizados en Inteligencia Artificial

Los lenguajes de programación contienen bibliotecas que ofrecen una amplia variedad de herramientas y opciones para entrenar modelos de IA, así como los recursos esenciales para llevar a cabo pruebas y validación del sistema.

La Tabla 5 muestra una comparación entre los lenguajes más utilizados en sistemas de inteligencia artificial. [12] [13] [14]

Tabla 5. Análisis de los lenguajes de programación empleados en el campo de la Inteligencia Artificial. [12] [13] [14]

Característica	Python	C++	Java	MATLAB
Aprendizaje Automático	Muy bueno, con herramientas como TensorFlow, Keras, Scikit-learn y PyTorch.	Excelente para aplicaciones de alto rendimiento con bibliotecas como Dlib y OpenCV	Buena, con bibliotecas como Weka y Deeplearning4j	Buena para prototipos rápidos con herramientas como MATLAB Machine Learning Toolbox
Procesamiento de Imágenes	Excelente, con OpenCV, Pillow y Scikit-image	Excelente, con OpenCV y otras bibliotecas de alto rendimiento	Buena, con bibliotecas como OpenCV y Java Advanced Imaging	Buena, con Image Processing Toolbox
Visión por Computadora	Excelente, con OpenCV, PyTorch y TensorFlow	Excelente, con OpenCV y Dlib	Buena, con OpenCV y BoofCV	Buena, con Computer Vision Toolbox
Redes Neuronales	Excelente, con TensorFlow, Keras y PyTorch	Excelente, con bibliotecas de alto rendimiento como Caffe y Darknet	Buena, con Deeplearning4j	Buena, con Neural Network Toolbox
Procesamiento de Datos	Excelente, con pandas, NumPy y Dask	Buena, pero menos utilizada que Python y R para estas tareas	Buena, con Apache Spark y Hadoop	Buena, con herramientas integradas para análisis y visualización de datos
Rendimiento	Moderado a alto, depende de las bibliotecas utilizadas y si se usa compilación Just-in-Time (JIT) con PyPy	Alto, excelente rendimiento y control de bajo nivel	Moderado a alto, con buen rendimiento en aplicaciones empresariales	Moderado, orientado a prototipos rápidos más que a aplicaciones de alto rendimiento

b. Plataformas para el desarrollo y la administración de datos en inteligencia artificial (IA)

Este tipo de plataformas permiten aplicar técnicas de inteligencia artificial par el análisis de datos agrícolas, la optimización de recursos y el incremento de la producción. La Tabla 6 presenta una comparación entre las identidades de los empleados en el ámbito de la inteligencia artificial y en el sector de la agricultura de precisión. [15]

Tabla 6. Comparación de Plataformas de desarrollo y gestión de datos [15]

Características	Google Colab	Amazon AWS (SageMaker)	Microsoft Azure ML
Modelo de uso	Basado en la nube, gratuito con límites	Basado en la nube, pago por uso	Basado en la nube, pago por uso
Facilidad de uso	Muy fácil, no requiere configuración local	Moderada, necesita configuración y gestión	Moderada, necesita configuración y gestión.
Lenguajes soportados	Python	Python, R, Julia, TensorFlow, PyTorch	Python, R, PyTorch, TensorFlow
Recursos de computación	GPUs y TPUs limitadas, gratis o pago por uso	Amplia variedad de instancias, incluyendo GPUs y TPUs	Amplia variedad de instancias, incluyendo GPUs
Almacenamiento de datos	Google Drive y almacenamiento temporal	Almacenamiento en S3 o EFS	Almacenamiento en Blob y otros servicios
Integración con Machine Learning	TensorFlow, PyTorch, Keras	SageMaker para entrenar, desplegar y manejar modelos	Azure ML para entrenamiento, despliegue y manejo de modelos.
Visualización de datos	Integrado con Matplotlib, Seaborn, Plotly	Integración con herramientas de visualización como SageMaker Studio	Integrado con Power BI y Azure Dashboards
Colaboración	Muy fácil compartir y colaborar en tiempo real (Google Drive)	AWS IAM para permisos, difícil de compartir en tiempo real	IAM y RBAC para permisos, difícil de compartir en tiempo real
Escalabilidad	Limitada (dependiente de la cuenta y recursos gratuitos o pagos)	Altamente escalable, desde instancias pequeñas hasta grandes clústeres	Altamente escalable con clústeres de máquinas virtuales
Costo	Gratuito con límite de recursos, opción de pago	Pago por uso de instancias y almacenamiento	Pago por uso de instancias y almacenamiento

Gracias a su capacidad para procesar grandes cantidades de datos y automatizar tareas, permiten a los investigadores y desarrolladores trabajar de manera más eficiente en proyectos que involucran agricultura de precisión, visión por computadora, y machine learning.

1.3.7 Métodos usados actualmente para estimar la cantidad de producción en la agricultura

La producción de mora ha sido tradicionalmente estimada mediante métodos manuales y encuestas, pero con la llegada de nuevas tecnologías, los sistemas de predicción han avanzado hacia el uso de modelos basados en machine learning. Este apartado analiza los métodos actuales, sus ventajas y limitaciones, tanto en enfoques tradicionales como tecnológicos. [16]

a. Métodos tradicionales

Los métodos tradicionales para predecir la cantidad de producción de mora incluyen encuestas a los agricultores, inspección visual en el campo y estimaciones manuales basadas en la experiencia y el conocimiento local. Estos métodos dependen en gran medida de la observación directa y el registro de información a mano, lo que genera variaciones en la precisión y la fiabilidad de los datos. [16]

b. Sistemas tecnológicos y modelos de machine learning aplicados

Con la introducción de nuevas tecnologías, la agricultura de precisión en Ecuador ha comenzado a integrar sistemas automatizados y modelos de machine learning para estimar la producción de mora. [16]

Estos modelos procesan grandes volúmenes de datos provenientes de imágenes satelitales, drones, sensores y otras fuentes, lo que permite hacer predicciones más precisas basadas en el análisis de variables como el clima, la densidad de plantas y la salud del cultivo. [16]

c. Ventajas y limitaciones de los métodos actuales en Ecuador

En la Tabla 7 se observa las ventajas, desventajas, limitaciones y la aplicabilidad que tienen en Ecuador de los diferentes métodos usados actualmente para predecir la cantidad de producción en la agricultura. [16]

Tabla 7. Ventajas y Desventajas de los métodos usados en Ecuador. [16]

Método	Ventajas	Limitaciones	Aplicabilidad en Ecuador
Encuestas y estimaciones manuales	Bajo costo, facilidad de implementación.	Alta subjetividad, baja precisión.	Ampliamente utilizado en áreas rurales.
Inspección visual	Dependencia del conocimiento local.	Elevado margen de error, costos de mano de obra altos.	Común en pequeñas parcelas.
Sistemas de visión artificial	Alta precisión, automatización, reducción de tiempo.	Requiere inversión inicial en tecnología	Se está implementando en proyectos de agricultura tecnificada.
Modelos de machine learning	Capacidad de analizar grandes volúmenes de datos, predicciones precisas.	Complejidad en la implementación, necesidad de datos históricos.	En desarrollo en áreas tecnificadas.

En la Tabla 7 se muestra información sobre las técnicas actualmente empleadas, con sus ventajas, limitaciones y aplicabilidad en Ecuador para la estimación de producción agrícola que son cuatro: encuestas y estimaciones manuales, inspección visual, sistemas de visión artificial y modelos de machine learning, por lo que permite analizar el estado actual de estas técnicas en el país, lo que conlleva a el cumplimiento del primer objetivo.

1.3.8 Visión Artificial en el monitoreo de cultivos

La visión artificial se ha convertido en una herramienta clave para el monitoreo eficiente de cultivos, permitiendo la recolección y análisis de datos visuales para optimizar las decisiones en agricultura de precisión. Esta tecnología posibilita la identificación temprana de problemas, como plagas o enfermedades, así como el seguimiento del crecimiento de las plantas y la estimación de la producción. [17]

Diversas técnicas y tecnologías de visión artificial han sido adoptadas para estas tareas, cada una con características que las hacen más o menos apropiadas según el tipo de cultivo y las necesidades específicas del entorno. [17]

En la Tabla 8 se presenta documenta diferentes tecnologías de visión artificial más utilizadas en la agricultura de precisión como: cámaras multiespectrales, cámaras RGB, cámaras hiperespectrales, sensores LiDAR y drones con visión artificial, especificando las aplicaciones, ventajas y desventajas de las tecnologías a utilizar, por lo que se cumple el segundo objetivo. [17]

Tabla 8. Comparación de las soluciones de visión artificial más utilizadas en la agricultura de precisión. [17]

Tecnología de Visión Artificial	Aplicación	Ventajas	Desventajas
Cámaras Multiespectrales	Detección de estrés hídrico, sanidad vegetal.	Permite capturar información en varias longitudes de onda.	Costo elevado, procesamiento complejo.
Cámaras RGB	Seguimiento de crecimiento, conteo de frutos.	Accesibles y fáciles de integrar en drones y robots.	Sensibilidad limitada a cambios en luz y clima.
Cámaras Hiperespectrales	Análisis avanzado de calidad de cultivos.	Alta precisión en la detección de variables físicas y químicas.	Alto costo, grandes volúmenes de datos a procesar.
Sensores LiDAR	Mapeo 3D del terreno y estructura de plantas.	Proporciona información detallada sobre la topografía y la biomasa.	Equipos costosos, difícil implementación en grandes áreas.
Drones con Visión Artificial	Monitoreo general de parcelas.	Alta eficiencia en la recolección de datos en grandes áreas.	Limitación por autonomía de vuelo y condiciones meteorológicas.

a. Configuración básica de un sistema de Visión Artificial

Los pilares del sistema físico de la visión artificial son el sistema de formación de las imágenes y el sistema de procesamiento de estas. En la Tabla 9 subsistema puede ser resueltas en las siguientes reflexiones introductorias: [18]

Tabla 9. Sistema físico de Visión Artificial. [17]

Sistema físico de Visión Artificial	
Subsistema de iluminación	Se encarga de proporcionar la luz necesaria para que los objetos a visualizar sean capturados correctamente. Esto puede incluir lámparas, pantallas fotográficas, filtros de luz, láseres y otros elementos.
Subsistema de captación	Transforman la luz reflejada por los objetos en señales eléctricas. Las cámaras CCD son las más comunes, pero también existen sensores para diferentes espectros, desde rayos gamma hasta microondas.
Subsistema de adquisición	Recibe las señales eléctricas de las cámaras y las convierte en señales de vídeo. Cada vez más, estas señales son digitales, pero aún se utilizan formatos analógicos como CCIR, PAL, RS170 y NTSC.
Subsistema de procesamiento	Generalmente un computador o un grupo de computadores (clúster), aplica algoritmos de visión artificial a las imágenes digitalizadas.
Subsistemas de periféricos	Reciben la información procesada por el sistema de visión artificial y la utilizan para interactuar con el mundo real. Esto puede incluir monitores para mostrar imágenes, robots para realizar tareas, impresoras para imprimir resultados o cualquier otro dispositivo que pueda utilizar la información de alto nivel.

En la Figura 8 se observa el esquema de los subsistemas físicos de un equipo utilizado en visión artificial, muestra cómo funciona por medio de una cámara hacia el computador u ordenador y sus diversos procesos.

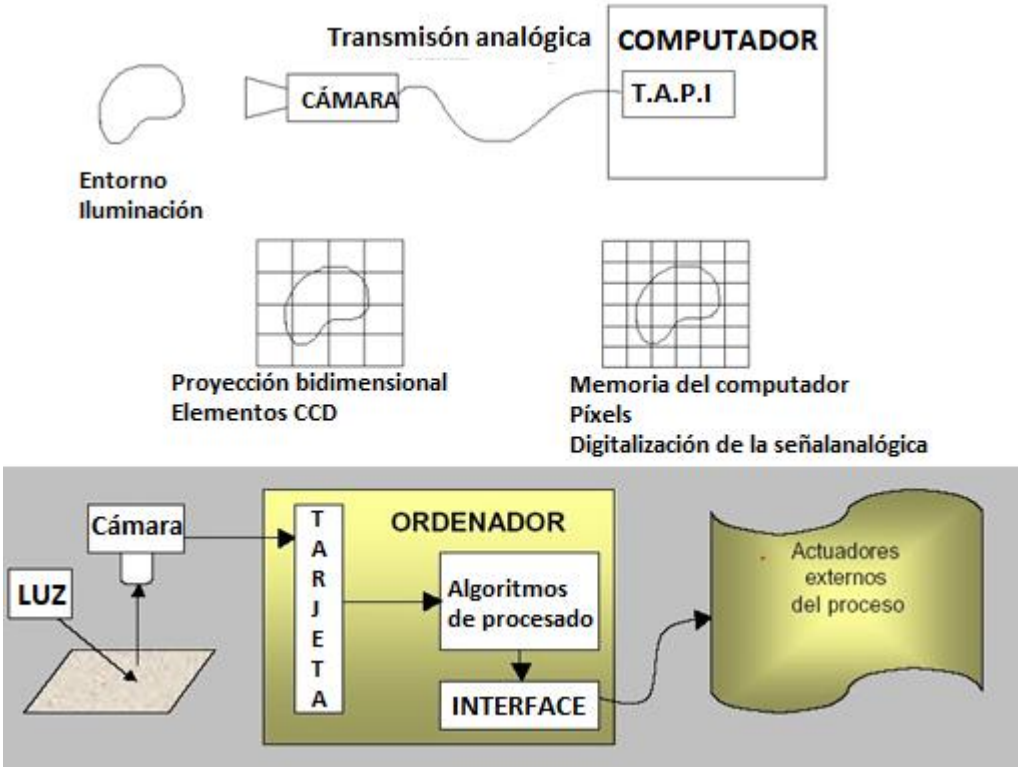

Figura 8. Subsistemas físicos de un equipo de visión artificial. [18]

El sistema básico de un sistema de visión por computadora empieza con una cámara tomando una fotografía después se procesa la información utilizando herramientas como CCD para una proyección bidimensional. [18]

b. Representación de las imágenes en los computadores digitales

Las imágenes que se procesan en el computador se capturan mediante una cámara de video y se almacenan en su memoria utilizando tarjetas de adquisición de video. Esta información visual se representa en forma de matriz bidimensional, donde cada elemento se denomina píxel. Para acceder a un píxel específico, se deben indicar sus coordenadas, es decir, la fila y la columna que ocupa dentro de la matriz. [18]

En la Figura 9 se observa que el valor de un píxel se representa mediante una función f(x, y), donde x es el índice de la fila e y es el índice de la columna. Esta función asigna

un valor numérico al píxel, que generalmente representa su brillo o intensidad de color. [18]

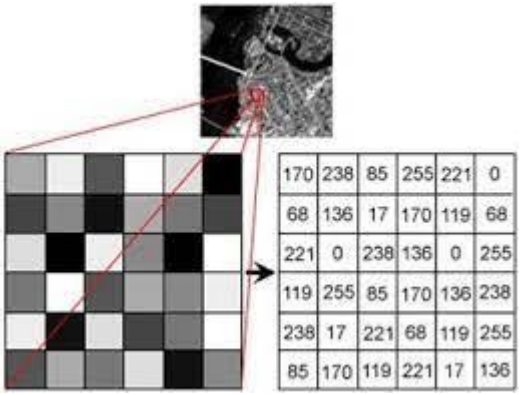

Figura 9. Organización matricial uniforme de una imagen digital. [18]

c. Etapas básicas de una aplicación en Visión Artificial

La primera etapa se centra en la adquisición de imágenes de la mejor manera posible. Para ello, se utilizan técnicas fotográficas como la iluminación adecuada, lentes apropiadas, cámaras de alta calidad, filtros y pantallas para resaltar las características visuales de los objetos que se desean analizar, como formas, texturas, colores y sombras. Un buen diseño en esta etapa inicial es crucial para el éxito de muchas aplicaciones de visión artificial. [18]

Una vez capturada la imagen, en la se procede a su preprocesamiento para mejorar su calidad informativa y facilitar el análisis posterior. En la Figura 10 se observa las etapas en donde se realizan diversas operaciones, como [18]:

- Reducción de ruido.
- Deconvolución.
- Regularización.
- Mejora del contraste.
- Optimización de la distribución de intensidad.
- Realce de características.

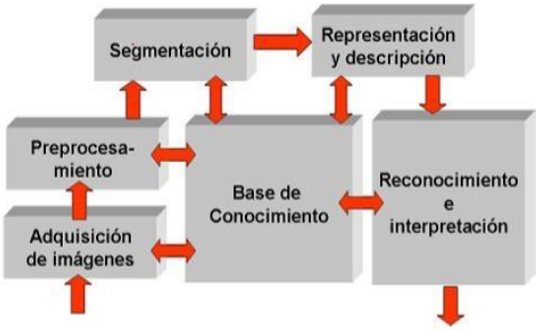

Figura 10. Etapas de una aplicación de visión artificial. [18]

d. Aplicaciones

Actualmente, las aplicaciones de la visión artificial abarcan un amplio espectro de áreas y continúan expandiéndose gracias a su enorme potencial y versatilidad. Este campo combina algoritmos avanzados, procesamiento de imágenes y aprendizaje automático para resolver problemas complejos y optimizar procesos en diversos sectores.

En la Tabla 10 se citan algunos de los campos donde es empleada esta disciplina. [18]

Tabla 10. Visión artificial y sus aplicaciones. [18]

Área de producción	Aplicación
Control de calidad	• Inspección de productos (papel, aluminio, acero, etc). • Identificación de piezas. • Etiquetados (fechas de caducidad). • Inspección de circuitos impresos. • Control de calidad de los alimentos.
Robótica	• Control de soldaduras. • Guiado de robots (vehículos no tripulados).
Biomédicas	• Análisis de imágenes de microscopia (virus, células, proteínas). • Resonancias magnéticas, tomografías, genoma humano.
Reconocimiento de caracteres	• Control de cheques. • Inspección de textos.
Astronomía	• Exploración del espacio.
Control de tráfico	• Matriculas de coches. • Trafico viario.
Meteorológica	• Predicción del tiempo.
Agricultura	• Interpretación de fotografías aéreas. • Control de plantaciones.
Militares	• Seguimiento de objetivos. • Vigilancia por satélite.

1.3.9 Vehículos Aéreos no tripulados para la adquisición de imágenes

La utilización de UAV en la agricultura de precisión ha incrementado la eficiencia en la captura de imágenes para el monitoreo y análisis de cultivos. Estos dispositivos permiten la obtención de datos visuales desde diferentes perspectivas y a gran escala, optimizando el uso de recursos y mejorando la toma de decisiones. Estas aeronaves que operan sin la necesidad de un piloto a bordo. [19]

a. Clasificación de UAVs

Los UAV se clasifican en diferentes categorías según su configuración, tamaño, alcance y aplicación. En la Figura 11 se muestra algunos de los tipos más comunes incluyen: [20]

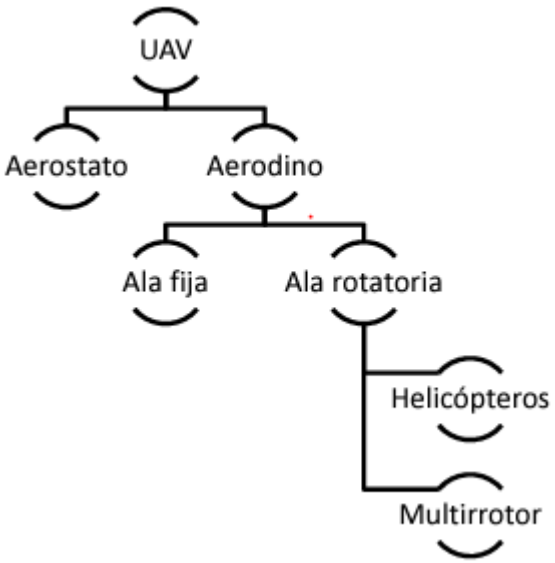

Figura 11. Clasificación de los UAVs. [20]

- ***Aerodino***

Los UAVs más empleados en la actualidad son los aerodinos, ya que poseen características que los hacen ideales para una amplia variedad de aplicaciones. Dentro de esta categoría, se destacan dos grandes grupos en función del tipo de ala que emplean: los de ala fija y los de ala rotatoria. [20]

- ✓ **Ala Fija**

Una de las principales ventajas de los UAV de ala fija en comparación con los de ala rotatoria es su notable autonomía. Similar a los aviones tradicionales, utilizan alas para generar sustentación aerodinámica. Ofrecen mayor alcance y velocidad que los multirrotor, pero son menos maniobrables. como se muestra en la Figura 12. [20]

Figura 12. Ala Fija. [20]

- ✓ **Ala rotatoria**

Una característica sobresaliente de los UAVs de ala rotatoria en comparación con los de ala fija es su capacidad para realizar despegues y aterrizajes verticales, además de mantener vuelos estacionarios. Según la cantidad de rotores o su configuración, estos UAVs pueden clasificarse en diferentes categorías. [20]

Helicópteros

Los helicópteros destacan por su notable capacidad de carga y autonomía, atribuida al uso de un único motor y una hélice de gran tamaño. En comparación con un cuadricóptero, consumen aproximadamente una cuarta parte de la energía. Además, son más eficientes desde el punto de vista aerodinámico como se muestra en la Figura 13 . [20]

Figura 13. Helicóptero. [20]

Multirrotores

Poseen varios rotores, generalmente cuatro o seis, que proporcionan sustentación y control direccional. Son populares para aplicaciones como fotografía aérea y videografía debido a su maniobrabilidad y estabilidad como se muestra en la Figura 14. [20]

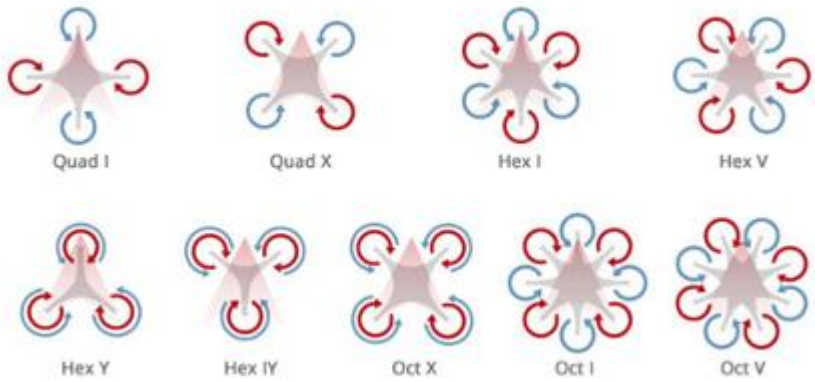

Figura 14. Multirrotor. [20]

b. Componentes de un UAV

Los componentes principales de un UAV incluyen: [19]

Marco: La estructura principal del UAV, que proporciona soporte a los demás componentes. [19]

Motores: Los propulsores que generan la fuerza necesaria para el vuelo. [19]

Hélices: Las palas giratorias que convierten la energía de los motores en empuje. [19]

Baterías: La fuente de alimentación del UAV, que proporciona energía para los motores, los sistemas electrónicos y la carga útil. [19]

Controladores de vuelo: Los microcontroladores que procesan datos de los sensores y envían señales a los motores para controlar el movimiento del UAV. [19]

Sensores: Dispositivos que recopilan información sobre el entorno del UAV, como su posición, orientación, altitud y velocidad. [19]

c. UAV disponibles para la captura de imágenes en cultivos

Hoy en día hay diversos modelos de dones o UAV diseñados o modificados para realizar diversas actividades que abarquen a temas de agricultura de precisión. Estos drones cuentan con cámaras integradas o módulos intercambiables que permiten capturar imágenes en diferentes espectros, como el visible (RGB), el infrarrojo cercano y el multiespectral, siendo fundamentales para el análisis de salud de los cultivos y la estimación de la producción. [19]

Algunos de los drones más populares en este ámbito son: [19]

- ✓ DJI Tello EDU
- ✓ DJI Mini 2
- ✓ DJI Phantom 3

d. Comparación de características clave de los UAV

A la hora de seleccionar un dron para la adquisición de imágenes en cultivos, es fundamental considerar características técnicas como la calidad de la cámara, la autonomía de vuelo y el precio, ya que estas influyen directamente en la capacidad del dron para capturar imágenes precisas y eficientes en grandes áreas de terreno. En la Tabla 11 se muestra la comparación de los drones utilizados para el conteo de frutos y la estimación de producción:

Tabla 11. Comparación de los drones utilizados para el conteo de frutos y la estimación de producción. [21] [22] [23]

Dron	Descripción	Cámara	Estabilizador	Programable	Autonomía de Vuelo	Peso	Compatibilidad
DJI Mini 2	Dron compacto y liviano con una cámara de alta calidad.	4K, 24 MP	3 ejes	Limitada (SDK)	31 min	249 g	Compatible con Python (SDK)
DJI Tello EDU	Dron programable para fines educativos, perfecto para aplicaciones básicas.	720p, 5 MP	Digital	Sí (SDK abierto)	13 min	87 g	Compatible con Python (SDK)
DJI Phantom 3	Dron resistente, diseñado para vuelos más prolongados y estables.	2.7K, 12 MP	3 ejes	Limitada	25 min	1.2 kg	Compatible con Python (SDK)

1.3.10 Planificación de trayectorias

El control de un UAV es importante para garantizar su estabilidad, navegación precisa y el cumplimiento de tareas específicas. Los algoritmos de control se diseñan considerando las dinámicas no lineales del UAV, las condiciones del entorno y las limitaciones del sistema como se muestra en la Figura 15. [24]

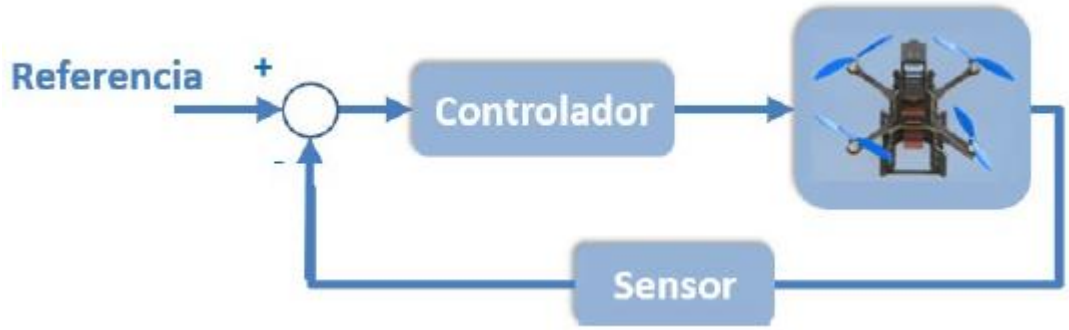

Figura 15. Diagrama de un sistema de control en lazo cerrado. [24]

a. Tipos de objetivos de control

Existen 3 tipos de objetivos de control: control punto a punto, seguimiento de trayectorias y evasión de obstáculos. [24]

- ***Control Punto a Punto***

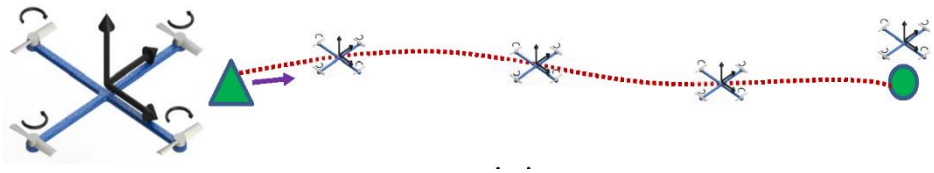

Figura 16. Control punto a punto. [24]

- ***Seguimiento de Trayectorias***

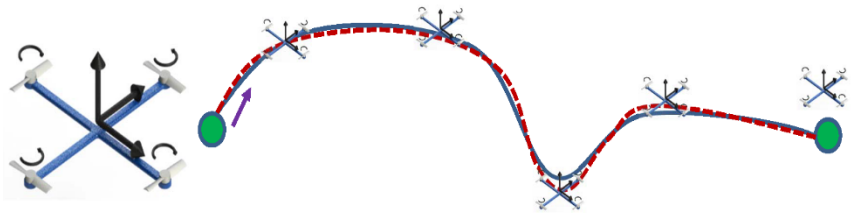

Figura 17. Seguimiento de trayectorias. [24]

- ***Evasión de Obstáculos***

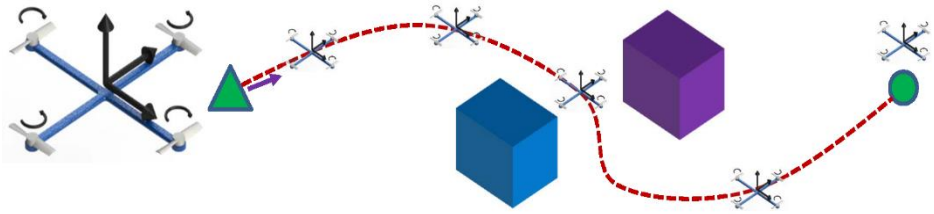

Figura 18. Evasión de obstáculos. [24]

b. Algoritmos de control

- ***Controladores lineales***

Los controladores lineales son utilizados en sistemas de control debido a que son muy sencillos, robustos y eficacia para aplicaciones prácticas incluyendo a vehículos aéreos

no tripulados, los controladores lineales se basan en ecuaciones lineales, que ayuda a simplificar el diseño. [24]

- ***Controladores no lineales***

Un controlador no lineal se considera cuando el sistema tiene variaciones dinámicas que no pueden representarse por medio de ecuaciones lineales. Por lo que las entradas y las salidas no son proporcionales, además los sistemas no lineales presentan respuestas complejas, presentan un mejor desempeño en cuando existen perturbaciones, por lo que este sistema es más complejo que un sistema lineal. [24]

En la Tabla 12 se muestra las estrategias de control que se aplican en cuadrotores. [24]

Tabla 12. Estrategias de control aplicadas a cuadrotores. [24]

Controladores lineales	**Controladores no lineales**
Control PD, PI, PID	PD robusto
Regulador Cuadrático Lineal (RCL)	Control Adaptable
Desigualdades matriciales lineales	Control Backstepping combinado con control H_∞
Realimentación de estados completa	Control por Modos Deslizantes (CMD)

c. Planificación de trayectorias con fotogrametría

Un plan de vuelo fotogramétrico es un archivo georreferenciado que define en detalle la trayectoria que debe seguir una aeronave no tripulada o tripulada para capturar imágenes aéreas. Además, en una ruta de vuelo especifica algunos parámetros importantes como la altitud, la configuración de la cámara o el láser, y cualquier otra variable relevante para garantizar la calidad y precisión de los datos obtenidos. [25]

- ***Factores principales en una planificación de vuelo fotogramétrico***

En la Tabla 13 se presentan diversos factores importantes para llevar a cabo una planificación de vuelo fotogramétrico como la velocidad, la altura, el campo de visión y la huella de registro.

Tabla 13. Factores clave en la planificación del vuelo. [25]

Altitud	La altitud se refiere a la distancia vertical en relación con el nivel del mar, mientras que la altura es la distancia entre la aeronave y el suelo. Ambas influyen en factores como las condiciones climáticas, la seguridad del vuelo y la resolución de las imágenes.
Velocidad	La velocidad de vuelo está directamente relacionada con la densidad de puntos en un levantamiento LiDAR. Velocidades más bajas permiten obtener una mayor densidad de puntos, pero aumentan el tiempo de misión.
Campo de visión (FOV) y Huella de Registro	El FOV, junto con la altura de vuelo, determina la huella de registro, es decir, el área del terreno cubierta por cada imagen. La huella de registro, calculada a partir del FOV y otros parámetros, indica la zona real cubierta por las fotografías.

- ***Ficheros necesarios para la planificación y ejecución del vuelo***

Una planificación de vuelo fotogramétrico precisa requiere no solo de un modelo digital del terreno (MDE), sino también de archivos vectoriales que delimiten el área de estudio y ubiquen puntos de interés. Estos archivos, al estar georreferenciados en el mismo sistema de coordenadas (WGS84), permiten verificar de manera visual y precisa que la planificación del vuelo cubra todas las zonas requeridas. Además, es fundamental considerar un margen de seguridad para asegurar que no se omita ninguna parte del área de interés, especialmente en zonas con irregularidades topográficas. [25]

- ***Pasos importantes para la planificación de trayectorias de Fotogrametría con Drones***

Los drones son los equipos ideales para obtener datos precisos y actualizados de manera rápida y segura. A diferencia de aviones o satélites, los drones ofrecen una mayor precisión, aunque su velocidad puede ser algo menor. Para crear modelos 3D detallados, utilizamos drones para capturar una serie de fotografías aéreas que luego procesamos con software especializado en la fotogrametría y planificación de trayectorias o misiones. [26]

Por lo que es importante tener en cuenta los siguientes pasos para realizar una buena planificación de trayectorias mediante un software que ayude a la planificación de trayectorias o también llamadas misiones. [26]

Delimitación el área de interés

Mediante el software Google Earth se puede cargar archivos geográficos y delimitar las zonas de interés tal como se muestra en la Figura 19. Esto nos permite estimar superficies, visualizar obstáculos y planificar rutas de vuelo de manera precisa. Combinando esta información con datos sobre el terreno y las condiciones de acceso, podemos optimizar nuestros vuelos. [26]

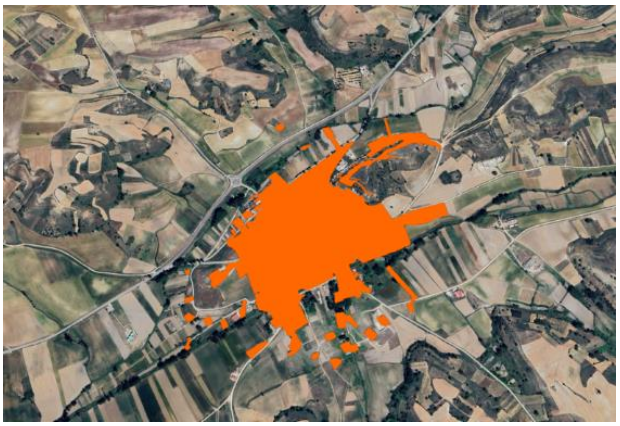

Figura 19. Delimitación del área. [26]

Elección del equipo

La selección del equipo adecuado es muy importante para el éxito de cualquier misión. Considerando las restricciones del área de vuelo y los requerimientos del cliente, por lo que se opta por equipos DJI tal como se muestra en la Figura 20. Estos drones tienden a manejarse con softwares intuitivos con facilidad de planificación de misiones, nos permiten operar de manera eficiente y segura. [26]

Figura 20. Selección del dron. [26]

Planificación de la misión

Para la planificación de trayectorias en general al utilizar los softwares o aplicaciones que ayudan a las misiones se tienen 2 opciones: la primera es el establecer las trayectorias y el segundo es de grabación de misiones. [26]

Establecer trayectorias crea una misión, primero se establece una ruta compuesta por varios puntos (waypoints). En la Figura 21 se observa los waypoints, además se pude definir acciones específicas como tomar fotografías o videos. Luego, se configura las opciones generales de la misión, como la altitud de vuelo, la velocidad y la orientación del dron. Finalmente, se asigna el dron y la cámara que se utilizarán para ejecutar la misión. [26]

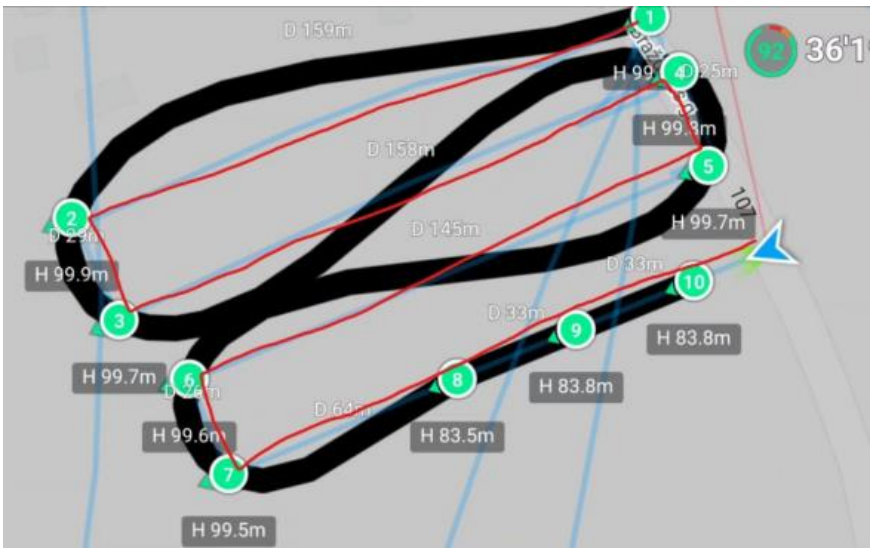

Figura 21. Waypoints con dron. [26]

La grabación de una misión consiste en pilotar el dron manualmente y registrar todas las acciones realizadas. Desde ascensos y descensos hasta movimientos de la cámara, cada detalle queda guardado. Al activar el sistema RTK que es una es una herramienta que mejora significativamente la precisión del posicionamiento y navegación, haciéndolos más eficientes y confiables, se garantiza una mayor precisión en la repetición de la misión. En la Figura 22 se muestra el control del dron mediante un usuario siendo poco autónomo el dron. [26]

Figura 22. Control de dron manualmente. [26]

d. DroneLink (Enlace de drones)

Dronelink es una plataforma de software que permite automatizar misiones complejas, como inspecciones y mapeos, y personalizar rutas de vuelo, esta herramienta ofrece un control mucho más preciso y eficiente. Compatible con una amplia gama de drones, Dronelink facilita la planificación de vuelos detallados y la integración con software de procesamiento de imágenes. En la Figura 23 se muestra la aplicación Dronelink. [27]

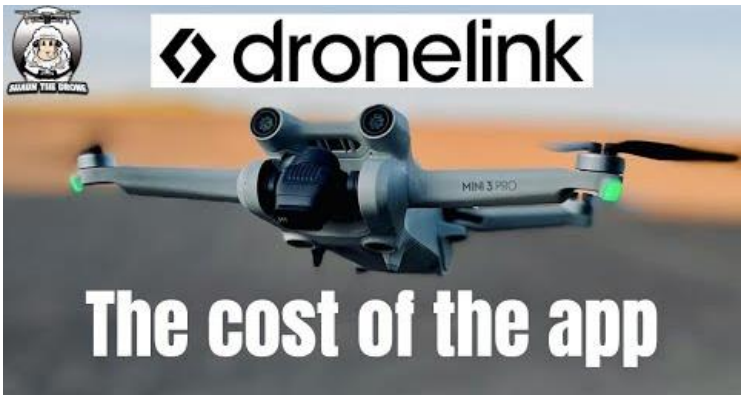

Figura 23. Aplicación Dronelink. [27]

e. Rainbow for DJI Drones (Arcoíris para drones DJI)

Rainbow es una aplicación de Android que lleva la experiencia de vuelo con drones DJI a un nivel superior. Se puede planificar rutas de vuelo precisas con hasta 99 waypoints, realizar fotografías panorámicas y seguir a objetos en movimiento de forma automática. Además, ofrece ajustes finos de la cámara y el dron, permitiéndote personalizar cada vuelo de acuerdo con las necesidades del usuario. La estabilidad y

fluidez de la imagen son inigualables, gracias a su avanzada tecnología de control de vuelo. En la Figura 24 se muestra la aplicación Rainbow for DJI Drones. [28]

Figura 24. Aplicación Rainbow for DJI Drones. [28]

f. Drone Harmony (Armonía de Drones)

Drone Harmony es una plataforma de software diseñada para optimizar y simplificar la planificación y ejecución de misiones de vuelo con drones. Esta herramienta se presenta como una alternativa más completa y flexible a los sistemas de control de vuelo estándar que vienen integrados en muchos drones, especialmente aquellos utilizados para tareas profesionales como inspecciones, topografía y agricultura. En la Figura 25 se muestra la aplicación Drone Harmony. [29]

Figura 25. Aplicación Drone Harmony. [29]

g. Comparación de características de aplicaciones para la planificación de trayectorias

Para escoger una aplicación de planificación de trayectorias o también llamadas misiones se debe tener en cuenta diferentes características tales como el tipo de dron que se tenga y su compatibilidad con la aplicación, la personalización de las rutas, precio y su facilidad de uso de la aplicación, por lo que en la Tabla 14 se muestra la comparación de 3 aplicaciones más conocidas en fotogrametría y en la planificación de trayectorias o misiones.

Tabla 14. Comparación de características de aplicaciones para la planificación de trayectorias. [27] [28] [29]

Características	Drone Link	Rainbow for DJI Drones	Drone Harmony
Compatibilidad con drones	Gran variedad de drones DJI	Principalmente drones DJI (Mavic, Phantom, Inspire, Mini)	Extensa selección de drones DJI, que abarca modelos profesionales.
Planificación de rutas	Sí, permite crear waypoints y rutas automáticas.	Sí, ofrece funciones avanzadas de waypoints y personalización de vuelo.	Sí, planificación automatizada de misiones, especialmente para inspecciones.
Características adicionales	Transmisión en vivo, control de la cámara, ajustes de vuelo.	Modo FPV, disparo panorámico, seguimiento de objetos, transmisión en vivo.	Inteligencia empresarial, generación de informes, integración con sistemas externos.
Facilidad de uso	Interfaz sencilla, ideal para principiantes.	Mayor personalización, puede ser más complejo para usuarios novatos.	Interfaz profesional, enfocada en automatización y análisis de datos
Precio	Depende de las opciones de suscripción.	Versión gratuita con opciones de suscripción.	Opciones de suscripción.
Orientado a	Usuarios casuales y profesionales	Fotógrafos y videógrafos que buscan mayor control.	Empresas que necesitan automatizar inspecciones y recopilar datos.

1.3.11 Algoritmos de machine learning usados en la predicción de producción.

El uso de algoritmos de machine learning en la predicción de producción agrícola ha transformado la manera en que los agricultores y científicos analizan grandes volúmenes de datos. Estas técnicas permiten crear modelos predictivos que procesan

información proveniente de diversas fuentes, como sensores, imágenes satelitales, drones y datos meteorológicos, para anticipar el rendimiento de los cultivos de manera más precisa. [11]

a. Regresión Lineal y Regresión Polinómica

La regresión lineal es uno de los algoritmos más simples y ampliamente utilizados para predecir la producción agrícola. Este modelo establece una relación entre una variable dependiente (rendimiento de los cultivos) y una o más variables independientes (factores como la humedad, temperatura o nutrientes del suelo). [28]

Regresión Lineal Simple: Utiliza una única variable independiente para hacer predicciones, como la relación entre la cantidad de agua suministrada y el rendimiento de una cosecha. En la Figura 26 se observa un ejemplo de regresión lineal simple. [30]

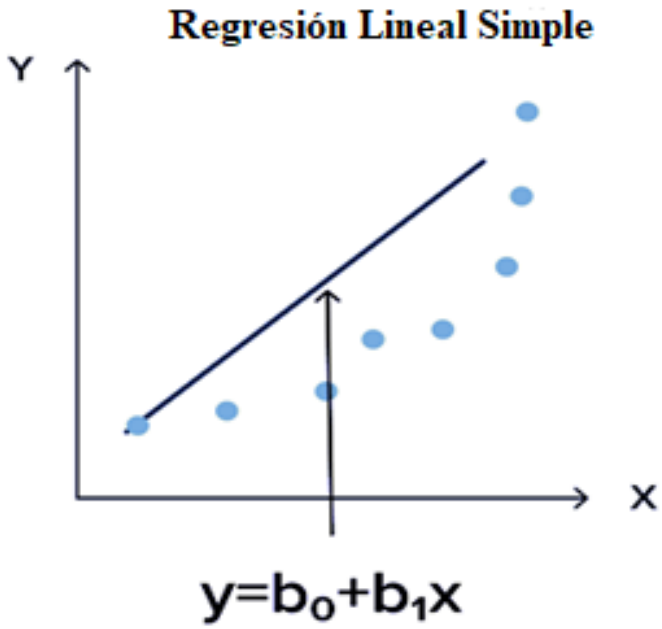

Figura 26. Regresión Lineal Simple utiliza una única variable independiente. [30]

Regresión Polinómica: Extiende la regresión lineal para capturar relaciones más complejas, modelando la influencia de múltiples factores ambientales y biológicos sobre la producción agrícola. En la Figura 27 observamos el comportamiento de los patrones en una regresión polinómica. [30]

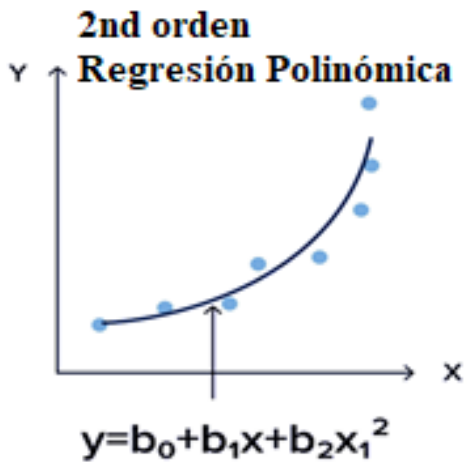

Figura 27. Comportamiento de los patrones en una regresión polinómica. [30]

b. Máquinas de Soporte Vectorial (SVM)

Las máquinas de soporte vectorial (SVM) son algoritmos potentes utilizados para clasificación y regresión en problemas no lineales. En el contexto de la agricultura, las SVM son utilizadas para predecir el rendimiento agrícola mediante la identificación de patrones complejos en los datos de entrada. [30]

SVM para Predicción de Producción: Este algoritmo es capaz de analizar grandes conjuntos de datos multivariados y detectar correlaciones entre variables difíciles de observar directamente, mejorando la precisión en las predicciones. En la Figura 28 observamos la correlación entre datos extensos. [30]

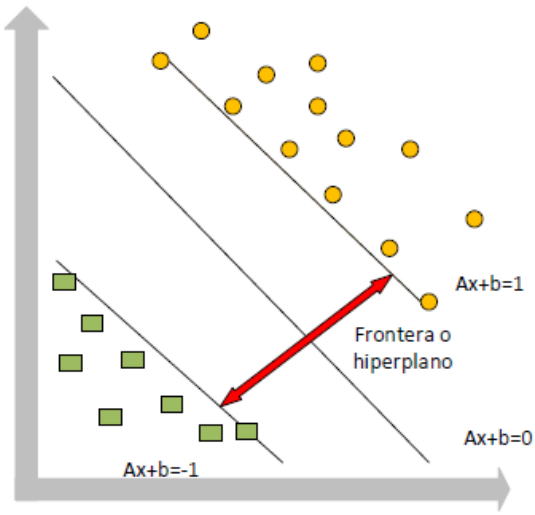

Figura 28. Correlación entre datos extensos. [30]

c. Árboles de Decisión y Bosques Aleatorios (Random Forest)

Los árboles de decisión y los bosques aleatorios son algoritmos de machine learning ampliamente usados para la predicción agrícola debido a su capacidad para manejar grandes volúmenes de datos y variables complejas. [30]

Árboles de Decisión: Estos modelos dividen los datos en subconjuntos más pequeños basados en condiciones predefinidas, creando un árbol de decisiones que se utiliza para hacer predicciones sobre la producción agrícola. En la Figura 29 observamos que son especialmente útiles para comprender cómo diferentes factores (como temperatura, fertilización y humedad) influyen en el rendimiento de los cultivos. [30]

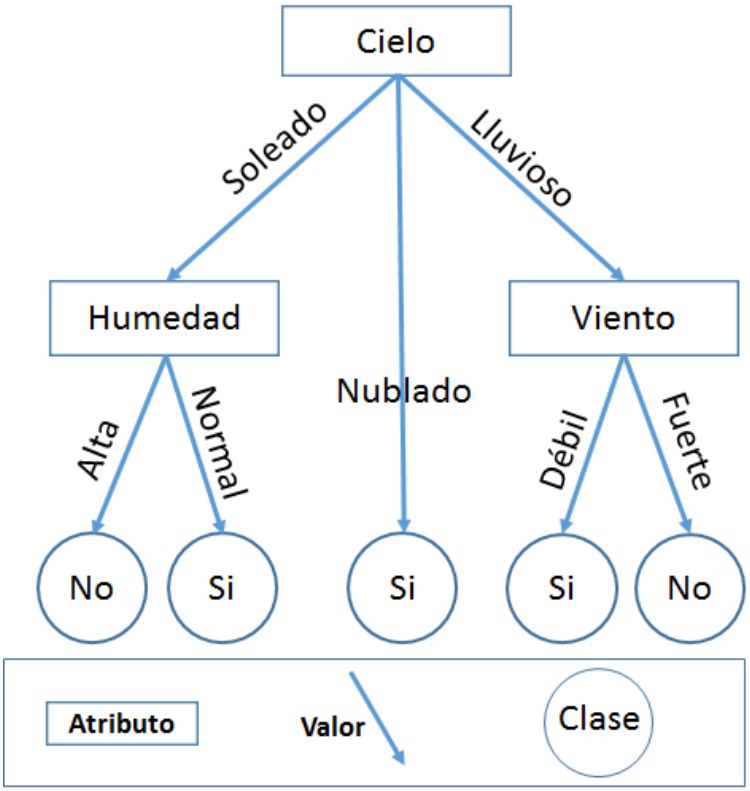

Figura 29. Árboles de Decisión. [30]

Bosques Aleatorios: Es una técnica de aprendizaje automático que utiliza varios árboles de decisión para hacer predicciones más precisas. En la Figura 30 los bosques aleatorios son especialmente robustos frente a la variabilidad en los datos y son muy efectivos en la predicción de rendimientos de cultivos. [30]

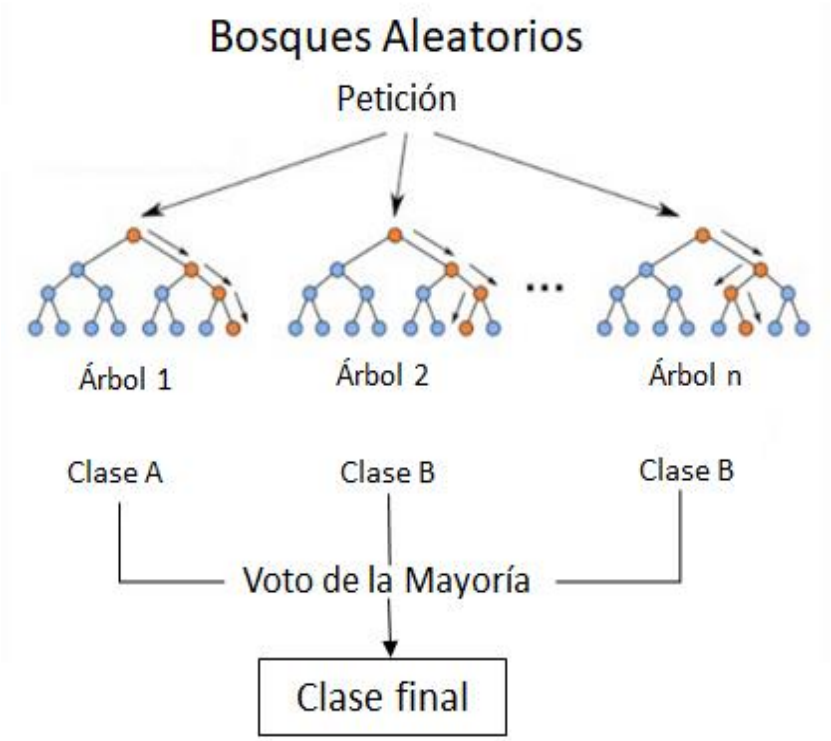

Figura 30. Bosques aleatorios. [30]

d. Algoritmos de Deep Learning

El Deep Learning es una rama avanzada de las redes neuronales que utiliza arquitecturas más complejas, como las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN). Estas técnicas son ideales para analizar imágenes y secuencias de datos temporales, proporcionando predicciones más precisas sobre la producción agrícola. [11]

Redes Neuronales Convolucionales (CNN): Se utilizan principalmente para el análisis de imágenes de cultivos, facilitando la estimación de la producción a partir de imágenes aéreas capturadas por drones. Estas redes son capaces de identificar patrones visuales complejos relacionados con el crecimiento de los cultivos y el estado de salud de las plantas. En la Figura 31 observamos la arquitectura de Redes Neuronales Convolucionales donde se observan cada una de las capas y el proceso que se realiza mediante una imagen como dato de entrada. [14]

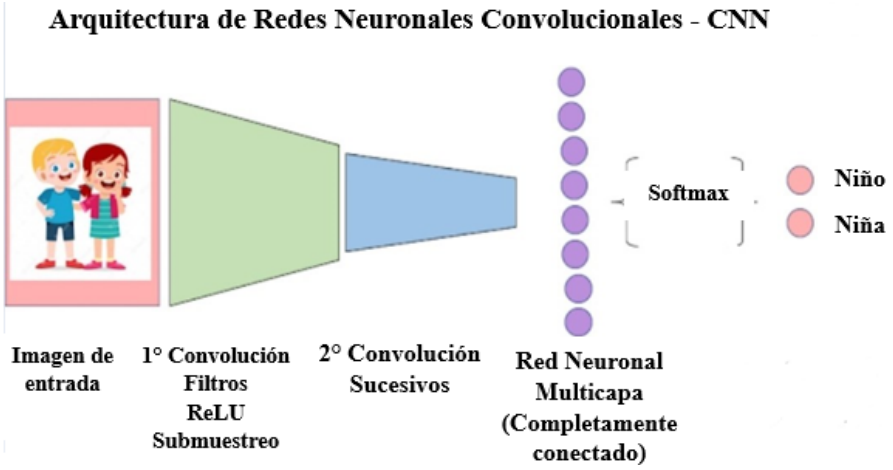

Figura 31. Arquitectura de Redes Neuronales Convolucionales [11].

Redes Neuronales Recurrentes (RNN): Son eficaces para manejar datos secuenciales y temporales, como las series de tiempo vinculadas al clima o al desarrollo del crecimiento de los cultivos. En la Figura 32 las RNN pueden modelar cambios en las condiciones agrícolas a lo largo del tiempo y predecir cómo estas influencias afectarán la producción. [11]

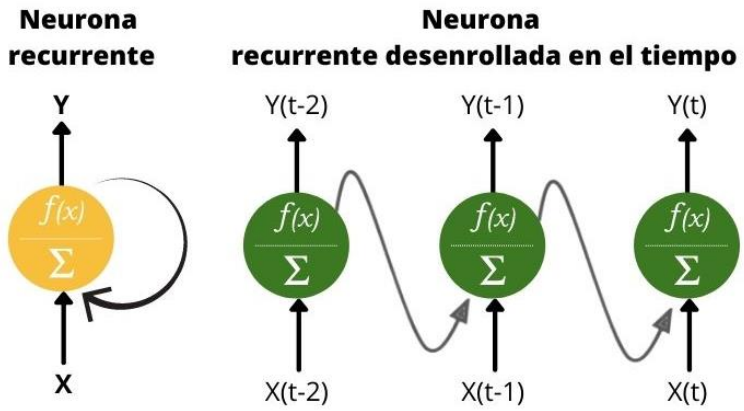

Figura 32. Redes Neuronales Recurrentes (RNN). [11]

e. Método de Montecarlo

Es una técnica matemática que predice los posibles resultados de un evento incierto, utiliza simulaciones aleatorias para dar soluciones a problemas matemáticos que suelen ser muy complejos o inciertos. Por lo que el método de Montecarlo es utilizado para examinar datos históricos y estimar posibles resultados futuros. [31]

Principios Básicos

- ✓ **Simulación Aleatoria:** Se generan muestras aleatorias en un rango para explorar posibles escenarios. [31]
- ✓ **Estadística y Probabilidad:** Se analizan los resultados de varias simulaciones para encontrar soluciones. [31]
- ✓ **Iteraciones Masivas:** Se repite el proceso múltiples veces para que sea más precisa la solución. [31]

Áreas de Aplicación

El método de Montecarlo es utilizado en varios ámbitos debido a su gran versatilidad como: [31]

- ✓ Cálculo de integrales complejas.
- ✓ Modelado financiero (valoración de opciones, riesgos).
- ✓ Física y química (simulaciones moleculares, dinámica de partículas).
- ✓ Inteligencia artificial y machine learning.
- ✓ Optimización de sistemas complejos.

Cómo Funciona el Método de Montecarlo

La simulación de Montecarlo funciona mediante la estimación de un conjunto de posibles resultados, por lo que se generan por un rango de valores proyectados y a su vez por un grupo de entradas fijas. El método realiza múltiples cálculos repetidos, utilizando grupos de números aleatorios que representan los valores de entrada. No tiene un límite preestablecido, ya que puede producir millas o incluso millones de resultados en distintos escenarios, lo que lo hace ideal para realizar predicciones tanto a corto como a largo plazo. [32]

El procedimiento de la simulación tiene los siguientes pasos:

- ✓ **Definición del Problema:** representación del sistema a analizar. [32]

- ✓ **Generación de Valores Aleatorios:** Se generan números aleatorios dentro de un rango determinado para las variables implicadas. [32]
- ✓ **Cálculo del Resultado:** Se calcula un resultado específico en función de las reglas establecidas para el problema. [32]
- ✓ **Promediado:** Los resultados obtenidos de todas las simulaciones se promedian, logrando así una estimación más precisa del comportamiento del sistema frente a distintas condiciones. [32]

Ventajas

Las principales ventajas del método de Montecarlo son las siguientes: [33]

- ✓ Es versátil y puede aplicarse a problemas complejos.
- ✓ Su manejo es eficiente.
- ✓ Permite evaluar múltiples escenarios e incertidumbres.
- ✓ Ideal para problemas donde las variables tienen comportamientos aleatorios.

Desventajas

Principales desventajas del método de Montecarlo: [33]

- ✓ Consume muchos recursos computacionales.
- ✓ La precisión depende del número de simulaciones.
- ✓ Puede ser lento para problemas con alta dimensionalidad.
- ✓ Si los datos históricos no son representativos, los resultados pueden ser inexactos.

1.3.12 Modelos y plataformas de inteligencia artificial para la agricultura

a. Redes neuronales para visión artificial en cultivos

Las redes neuronales artificiales son herramientas indispensables para el desarrollo de sistemas de visión artificial en la agricultura, permitiendo analizar y clasificar diferentes tipos de cultivos con alta precisión. En particular, las redes neuronales convolucionales (CNN) han ganado relevancia debido a su capacidad para procesar imágenes y extraer características significativas de los cultivos, permitiendo un

análisis automatizado y preciso. Aquí presentamos los modelos preentrenados y la adaptación de CNN para predicción en la agricultura de precisión. [34]

b. Modelos preentrenados aplicados en la clasificación de cultivos

Entre los modelos preentrenados más utilizados para la visión artificial en cultivos, la serie YOLOv8 (You Only Look Once versiones 8) ha demostrado ser especialmente efectiva en tareas de clasificación y detección de objetos debido a su rapidez y precisión. El Modelo VGG16, aunque es preciso en clasificación de imágenes, su velocidad lo hace menos práctico para aplicaciones en tiempo real como el monitoreo desde drones. En la Tabla 15 se presenta una comparación entre estos 3 modelos preentrenados aplicadas a la clasificación y análisis de cultivos: [34]

Tabla 15. Comparación entre YOLOv8 y VGG16 aplicadas a la clasificación y análisis de cultivos. [34]

Modelo	Funcionalidad Principal	Aplicación en Agricultura	Precisión	Velocidad (FPS)	Comentarios
YOLOv8	Identificación instantánea y exacta de objetos	Clasificación general de cultivos, detección de frutos o plagas	Alta	Muy rápida	Adecuado para tareas de detección de objetos múltiples, como conteo de frutos en plantas.
VGG16	Red neuronal convolucional profunda (clasificación de imágenes)	Detección de enfermedades en cultivos, análisis de calidad	Alta	Lenta	No se encuentra optimizado para la identificación de objetos en tiempo real. Es más lento, pero su arquitectura puede proporcionar alta precisión en imágenes estáticas.

c. Adaptación de redes neuronales convolucionales (CNN) para la predicción

Las redes neuronales convolucionales (CNN) son especialmente efectivas para el análisis de imágenes en agricultura debido a su capacidad para extraer y procesar características espaciales en múltiples niveles de abstracción. Las CNN no solo se utilizan para la clasificación de cultivos, sino también para la predicción de variables importantes, como el rendimiento de las plantas y su salud a lo largo del tiempo. [34]

- ***Estructura de una CNN adaptada para predicción***

Las CNN aplicadas en la predicción agrícola generalmente siguen una arquitectura similar, compuesta por varias capas de convolución seguidas de capas de agrupamiento (pooling) y capas completamente conectadas. Las capas de convolución detectan características clave en las imágenes, como la textura de las hojas o los patrones de crecimiento, mientras que las capas finales realizan la predicción basada en estas características. Estas redes pueden predecir el rendimiento de cultivos analizando cambios en las imágenes a lo largo de diferentes etapas del crecimiento. [34]

- ***Transferencia de aprendizaje para predicción***

Un enfoque común es utilizar una red preentrenada y aplicar transfer learning para adaptar el modelo a tareas específicas de predicción en cultivos. Esto se logra entrenando únicamente las capas superiores de la red para que reconozcan los patrones únicos de un tipo de planta o una variedad de cultivo. Por ejemplo, una CNN preentrenada en imágenes genéricas puede ser ajustada para predecir el rendimiento de un cultivo de frutas, basándose en imágenes que muestren el tamaño, la coloración y la densidad de los frutos. [34]

- ***Predicción de rendimiento***

Las CNN también pueden predecir el rendimiento agrícola a partir de imágenes de drones o satélites, analizando la densidad de las plantas, el estado del follaje y otros factores visuales. Con suficientes datos históricos, las redes pueden anticipar el comportamiento del cultivo en diferentes condiciones climáticas o ante la presencia de plagas, lo que facilita la planificación y el manejo eficiente del terreno agrícola. [34]

En esta sección se muestra modelos de inteligencia artificial muy utilizados, la información sobre modelos y técnicas específicas (YOLOv8, VGG16, CNN) aplicables a la detección del estado y cantidad de frutos en la planta de mora son logaritmos necesarios para detectar dichas características, además en la Tabla 14 nos muestra diferentes aplicaciones que utilizan algoritmos para la planificación de trayectorias mediante el UAV, con esta información detallada se cumple con el tercer objetivo.

1.3.13 Microordenador

Un microordenador o microcomputadora es un tipo de computadora que se basa en un microprocesador como unidad central de procesamiento. Las microcomputadoras se caracterizan por su tamaño compacto y su capacidad de procesamiento. [35]

a. NVIDIA Jetson Nano

Es una computadora bien pequeña o un módulo de procesamiento compacto y eficiente en el uso de energía desarrollado por NVIDIA, diseñado específicamente para manejar tareas de inteligencia artificial (IA) y computación en el borde (edge computing). Este dispositivo puede llevar a cabo inferencias de aprendizaje profundo en tiempo real, visión por computadora y sistemas autónomos. En la Figura 33 se muestra la NVIDIA Jetson Nano [36]

Figura 33. NVIDIA Jetson Nano. [36]

b. Raspberry Pi 5

La Raspberry Pi 5 es la última versión de microcomputadora de bajo costo de la Fundación Raspberry Pi, diseñada para una gran variedad de usos, desde proyectos educativos hasta aplicaciones industriales e IoT. Esta nueva versión presenta un rendimiento mejorado gracias a su procesador ARM Cortex-A76 de cuatro núcleos a 2,4 GHz y su GPU VideoCore VII, que soporta gráficos en alta definición y decodificación 4K, ideal para medios y aplicaciones avanzadas. En la Figura 34 se muestra la Raspberry Pi 5. [37]

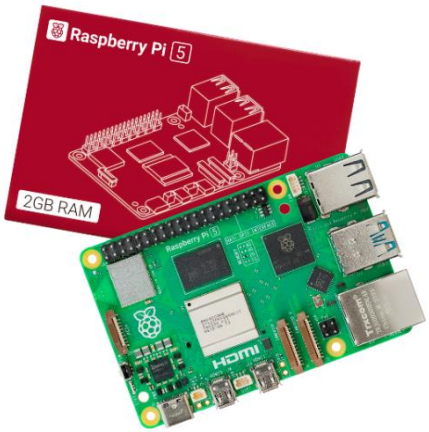

Figura 34. Raspberry Pi 5. [37]

c. Intel Neural Compute Stick

Es un dispositivo USB de bajo consumo que ofrece potencia de procesamiento para inteligencia artificial (IA) y aprendizaje profundo en diversas aplicaciones. Está pensado para desarrolladores, investigadores que buscan añadir capacidades de inferencia de redes neuronales a sus dispositivos sin necesidad de invertir en hardware especializado o costoso. Este dispositivo resulta perfecto para aplicaciones de computación en el borde, como visión artificial y análisis en tiempo real. En la Figura 35 se muestra el Intel Neural Compute Stick. [38]

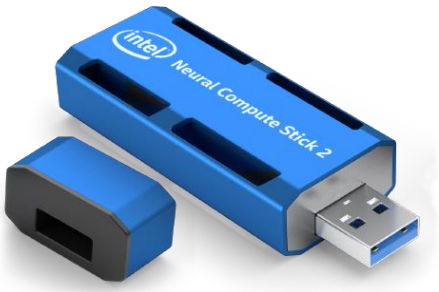

Figura 35. Intel Neural Compute Stick. [38]

d. Comparación de características de microcomputadores

Al elegir un microcontrolador para proyectos de adquisición de datos en el campo de la tecnología agrícola, es esencial evaluar características técnicas clave como la

capacidad de procesamiento, el consumo de energía y el costo. Estos aspectos determinan la efectividad del microcontrolador para gestionar y procesar datos de manera precisa y eficiente, especialmente en aplicaciones de monitoreo en grandes áreas de terreno.

En la Tabla 16 se muestra la comparación de 3 microcomputadores que más se utilizan para la ejecución de sistemas de proyectos con IA en la agricultura.

Tabla 16. Comparación de microcomputadores. [36] [37] [38]

Características	NVIDIA Jetson Nano	Raspberry Pi 5	Intel Neural Compute Stick
Procesador	ARM Cortex-A57 (4 núcleos)	ARM Cortex-A76 (4 núcleos)	N/A (requiere host)
GPU	NVIDIA Maxwell (128 núcleos CUDA)	VideoCore VI	Intel Movidius Myriad X
Memoria RAM	4 GB	8 GB	N/A
Almacenamiento	MicroSD, M.2 NVMe (opcional)	MicroSD, eMMC	N/A
Eficiencia energética	Alta	Media	Alta
Soporte para IA	Sí (CUDA, TensorRT, PyTorch)	Limitado	Sí (Deep Learning)
Compatibilidad con cámaras	CSI, USB	CSI, USB	Depende del host

1.3.14 Sistema de Gestión de Base de Datos (SGBD)

Un SGBD es un software que proporciona una interfaz amigable para interactuar con bases de datos, permitiendo operaciones como consultar, insertar, actualizar y eliminar datos.

a. Componentes de un SGBD

En la Tabla 17 se muestra varios componentes integrados que realizan tareas de gestión de datos:

Tabla 17. Componentes importantes de una SGBD. [39]

Motor de almacenamiento	Es la parte principal del SGBD que se comunica con el sistema de archivos del sistema.
Lenguaje de consulta	Incluyen el lenguaje SQ integrado y el lenguaje MQL de MongoDB, que permiten la interacción con bases de datos.
Procesador de consultas	Trata de definir las consultas de los usuarios y traducir reglas para comprender la base de datos.
Optimizador de consultas	Optimiza el rendimiento de las consultas y de la base de datos en general.
Catálogo de metadatos	Funciona como un repositorio de los elementos de la base de datos. Cada vez que se crea un nuevo objeto, este se registra en el catálogo, el cual también sirve para verificar solicitudes y brindar información sobre la estructura de la base de datos.
Administrador de registros	Registra las acciones del usuario, como inicios de sesión, entradas y otras operaciones realizadas en el SGBD.
Herramientas de informes y seguimiento	Incluyen herramientas para generar informes y monitorear el uso de recursos dentro del SGBD.
Utilidades de datos	Son herramientas adicionales para realizar copias de seguridad, recuperación, validación y reparación de datos, además de asegurar la integridad de los datos en servidores privados o compartidos.

b. Tipos de gestión de bases de datos

Los SGBD se clasifican según la distribución de datos, el modelo de datos utilizado y la cantidad de usuarios que puede manejar. A continuación, se presentan los tres tipos principales: [39]

- ***Sistema de gestión de bases de datos relacionales (SGBDR)***

Un sistema de gestión de bases de datos relacionales (SGBDR) es de naturaleza relacional y almacena datos estructurados en tablas con relaciones definidas. Utilizan el lenguaje SQL para interactuar con la base de datos. [39]

A continuación, sus principales características: [39]

- ✓ **Estructura:** Organizan los datos en tablas.
- ✓ **Usuarios:** Permiten el acceso simultáneo de múltiples usuarios.
- ✓ **Programas:** Gestionan las relaciones entre las tablas de datos.
- ✓ **Datos:** Están diseñados para manejar grandes volúmenes de datos.
- ✓ **Bases de datos distribuidas:** A diferencia de los SGBD, los SGBDR admiten bases de datos distribuidas.
- ✓ **Redundancia de datos:** Utilizan claves e índices para reducir la redundancia, característica que generalmente no poseen los SGBD.

- ***Sistema de gestión de bases de datos documentales***

Los sistemas de gestión de bases de datos de documentos almacenan datos en archivos JSON sin un formato unificado. Para realizar operaciones como lectura y escritura en la base de datos, generalmente emplean el lenguaje de consulta de MongoDB. [39]

- ***Sistema de gestión de bases de datos columnares***

Los sistemas de gestión de bases de datos basados en columnas organizan los datos en grupos para mejorar el rendimiento. Este formato permite un procesamiento analítico ágil, ya que facilita la lectura eficiente de los datos. Por ello, las empresas que almacenan grandes cantidades de datos en discos optan por estos sistemas. [39]

c. SQLite

SQLite es una herramienta de código abierto diseñada para almacenar datos en dispositivos integrados de manera simple, eficiente, potente y rápida, incluso en equipos con recursos de hardware limitados, como una PDA o un teléfono móvil. En la Figura 36 se muestra el icono característico de SQLite. [40]

Figura 36. Base de datos SQLite. [40]

d. PostgreSQL

Se trata de un sistema de gestión de bases de datos relacional de código abierto que admite SQL para consultas relacionales y JSON para consultas no relacionales. sobresale por su capacidad para manejar grandes volúmenes de datos y su adaptabilidad en cuanto a la posibilidad de expansión. PostgreSQL ha evolucionado hasta convertirse en uno de los sistemas de bases de datos más sofisticados y fiables, adecuado tanto para aplicaciones de pequeño alcance como para sistemas críticos en entornos empresariales. En la Figura 37 se observa el icono característico de PostgreSQL. [41]

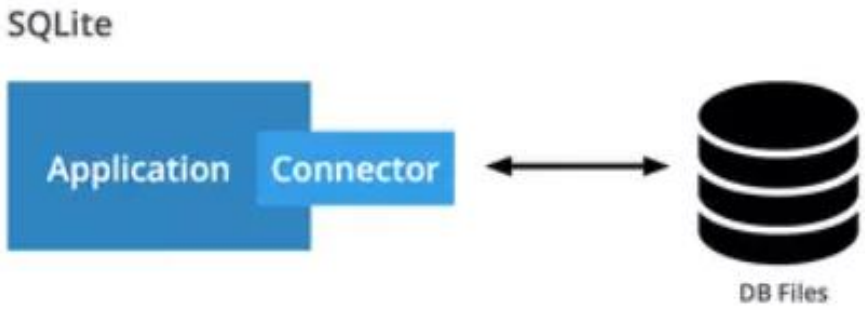

Figura 37. Base de datos PostgreSQL. [41]

e. MongoDB

Es un sistema de administración de bases de datos NoSQL que se fundamenta en documentos, estructurando la información en colecciones y documentos. Entre sus principales características destacan el manejo de datos no estructurados, el soporte para indexación avanzada y la replicación a través de APIs. En la Figura 38 se muestra la estructura de MongoDB. [42]

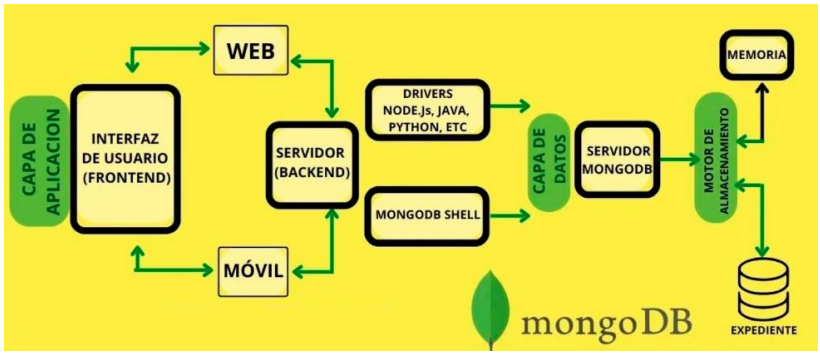

Figura 38. Administrador de base de datos MongoDB. [42]

f. Análisis comparativo de las características de los Sistemas de Gestión de Bases de Datos.

Al elegir un Sistema de Gestión de Base de Datos (SGBD), es importante tener en cuenta factores técnicos como la capacidad de almacenamiento, la rapidez de procesamiento y los costos involucrados, ya que influyen directamente en la eficiencia y exactitud al gestionar grandes cantidades de datos. Estos elementos son fundamentales para garantizar que el SGBD pueda gestionar y procesar información de forma óptima en aplicaciones que demandan velocidad y confiabilidad.

Tabla 18. Comparación de sistema de gestión de base de datos. [40] [42] [41]

Características	SQLite	MySQL	PostgreSQL
Tipo	Base de datos ligera embebida, local.	Sistema de gestión de bases de datos relacional (RDBMS).	Sistema de gestión de bases de datos relacional extendida (ORDBMS).
Almacenamiento de datos	Archivo local de base de datos	Almacenamiento en disco, opción de nube	Almacenamiento en disco, opción de nube
Facilidad de configuración	Muy fácil, no requiere configuración de servidor	Requiere configuración de servidor, pero tiene varias opciones simplificadas	Más compleja de configurar, pero ofrece mejor rendimiento y flexibilidad
Arquitectura	Sin servidor (embebida)	Basada en servidor	Basada en servidor
Tamaño de la base de datos	Adecuado para pequeños datasets	Apto para datasets medianos y grandes	Apto para grandes datasets, especialmente para análisis de datos
Licencia	Gratis (Public Domain)	Gratis (Licencia GPL), opciones comerciales disponibles	Gratis (Licencia PostgreSQL), también tiene opciones comerciales
Seguridad	Seguridad básica basada en permisos de archivo	Seguridad avanzada con autenticación de usuarios y permisos.	Seguridad avanzada con roles, autenticación y permisos a nivel de fila

En la Tabla 18 se presenta una comparación entre tres diferentes sistemas de gestión de bases de datos: SQLite, MySQL y PostgreSQL. La comparación se basa en las características más importantes de cada una de las bases de datos tales como su seguridad, arquitectura, escalabilidad y su almacenamiento.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema de estimación de producción del cultivo de mora empleando Visión Artificial y Machine Learning.

1.4.2 Objetivos específicos

- Determinar las técnicas empleadas actualmente para la estimación de producción de mora.
- Documentar las tecnologías y técnicas disponibles en Visión Artificial y Machine Learning aplicables al sistema de estimación de producción.
- Determinar algoritmos de visión artificial para la detección del estado y cantidad de fruto en la planta de mora y planificación de trayectorias mediante un UAV.
- Desarrollar una interfaz gráfica que interactúe con el sistema de automatizado de estimación producción de mora.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

Para el desarrollo del proyecto de investigación se han empleado diversos dispositivos y herramientas relacionadas con la inteligencia y visión artificial, seleccionados para satisfacer las necesidades técnicas del sistema. El sistema se ha diseñado para estimar la producción de cultivo de mora utilizando el lenguaje de programación Python y la arquitectura YOLOv8m para la detección de frutos de mora. El etiquetado de los datos se llevó a cabo mediante la herramienta Labelme, generando un conjunto de datos personalizado que incluye imágenes de moras en diferentes estados de madurez verde, rojas, y maduras o moradas.

El modelo fue entrenado con los datos recopilados, empleando Google Colab, aprovechando su capacidad de procesamiento en la nube, y se utilizó PyTorch como framework para el desarrollo y entrenamiento del modelo. Adicionalmente, la detección y clasificación en tiempo real se optimizó mediante el uso de una NVIDIA Jetson Nano, que actúa como unidad de procesamiento en el campo.

Las imágenes se capturaron con un dron DJI Mini 2 equipado con una cámara 4K, configurada para obtener tomas de alta calidad desde diferentes ángulos y condiciones ambientales. El dron realiza trayectorias planificadas para garantizar una cobertura completa de los cultivos, facilitando la adquisición de datos precisos y consistentes.

Para el manejo y almacenamiento de datos procesados, se implementó una base de datos MongoDB, lo que permite la gestión de la información generada por el sistema. En la Tabla 19, se detallan las características del hardware seleccionado, mientras que en la Tabla 20 se presenta el software empleado en el sistema.

Tabla 19. Características del hardware seleccionado.

Material	Características	Descripción
NVIDIA Jetson Nano	4 GB RAM, GPU de 128 núcleos CUDA	Procesamiento en tiempo real para visión artificial
DJI Mini 2	Cámara integrada 4K, estabilización avanzada	Captura de imágenes aéreas de alta resolución
Cámara Steren 4K	Resolución 1024p, >60 fps	Sensor de alta calidad para imágenes precisas

Tabla 20. Características del software empleado.

Software	Características	Descripción
Python	Lenguaje de programación	Desarrollo de algoritmos de visión artificial y análisis
YOLOv8m	Red neuronal convolucional para detección rápida	Detección de moras en diferentes etapas de madurez
PyTorch	Framework de aprendizaje profundo	Entrenamiento y ajuste del modelo YOLO
Labelme	Herramienta de etiquetado de datos	Creación de dataset personalizado
MongoDB	Base de datos NoSQL	Almacenamiento de datos procesados
Google Colab	Plataforma de cómputo en la nube	Entrenamiento del modelo con capacidad de GPU

2.2 Métodos

2.2.1 Modalidad de la investigación

Para el desarrollo del proyecto se aplicaron diversas modalidades de investigación, descritas a continuación:

Este proyecto tiene características de una investigación aplicada, ya que se emplearon conocimientos preexistentes en el campo de la visión artificial, machine learning y la

agricultura de precisión, con el objetivo de desarrollar un sistema para la estimación de la producción del cultivo de mora. Se enfoca en generar soluciones prácticas a problemas como la predicción de rendimiento.

La fundamentación del proyecto se realizó mediante una investigación bibliográfica basada en la recopilación de información proveniente de fuentes confiables, como libros, artículos científicos, datasheets de los dispositivos utilizados, publicaciones en internet y tesis relacionadas con sistemas de inteligencia artificial aplicados a la agricultura. Esta modalidad permitió formar una base teórica sólida para el desarrollo y validación del sistema.

Además, el proyecto también incorpora características de la investigación experimental y de campo. Investigación experimental ya que se ha diseñado y realizado experimentos para validar el funcionamiento del sistema de estimación. Esto incluyó el entrenamiento y evaluación del modelo de detección basado en YOLOv8m, la captura de imágenes con el dron DJI Mini 2, y la implementación de simulaciones para analizar el impacto de las pérdidas en la producción. La experimentación fue clave para ajustar el sistema y asegurar su precisión en condiciones controladas y la investigación de campo en la que se ejecutaron pruebas directamente en los cultivos de mora, donde el dron capturó imágenes reales de los cultivos en distintas etapas de madurez. Esto permitió evaluar el rendimiento del sistema en un entorno real, enfrentando condiciones ambientales y variaciones propias del campo.

2.2.2 Recolección de información

La recolección de información se realizó mediante el uso de libros, revistas, fuentes online y proyectos desarrollados, así mismo se tomó como referencia de datos estadísticos del Ecuador de producción de mora, con la finalidad de aporte para el desarrollo del proyecto. En la Tabla 21 se muestra los criterios de recolección de información que se utilizó para la investigación.

Tabla 21. Criterios de recolección de información.

Criterio	Descripción
¿Para qué?	Determinar el estado actual de la estimación de producción de mora.
¿De qué personas u objetos?	Cultivo de mora.
¿Como?	A través de la evaluación del sistema de estimación de producción de cultivo de mora.
¿Con que?	Utilizando artículos, libros, proyectos previos y datos estadísticos.
¿Cuándo?	En el periodo académico septiembre 2024 – febrero 2025
¿Donde?	En la provincia de Tungurahua, cantón Ambato, sector Montalvo.

2.2.3 Procesamiento y análisis de datos

Con la información recopilada se procedió de la siguiente manera:

• Recolección de la información.

• Revisión de la información obtenida.

• Análisis de la fundamentación teórica de la información sobre las variables de investigación.

• Deducción de la información más relevante y útil para el proyecto de investigación.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

3.1 Análisis y discusión de los resultados

Para el desarrollo del sistema de estimación de producción de cultivo de mora, se utilizó una Jetson Nano como plataforma principal, junto con el dron DJI Mini 2, enfocándose en la detección y clasificación de moras en tres estados de madurez: verdes, rojas y maduras. Este sistema combina algoritmos de visión artificial y redes neuronales para procesar imágenes capturadas desde el dron, identificando las características visuales específicas que permiten categorizar cada fruto según su grado de desarrollo.

La planificación de trayectorias gracias a la aplicación Rainbow for DJI Drones fue importante para la automatización del dron y su trayectoria en los cultivos de mora. Utilizando el DJI Mini 2, se definieron rutas de vuelo óptimas que garantizaron una cobertura completa y eficiente del área de cultivo. Estas trayectorias fueron establecidas para mantener de 1m hasta 2m máximo y velocidad de 0.3m/s, maximizando la calidad de las imágenes capturadas y minimizando las interrupciones debidas a condiciones climáticas adversas. Jetson Nano como plataforma principal, enfocándose en la detección y clasificación de moras en tres estados de madurez: verdes, rojas y maduras. Este sistema emplea algoritmos de visión artificial y redes neuronales para procesar imágenes del cultivo, identificando las características visuales específicas que permiten categorizar cada fruto según su grado de desarrollo.

Aunque los resultados obtenidos son prometedores, se recomienda realizar pruebas adicionales en condiciones climáticas diferentes como lluvias y vientos fuertes en el cultivo para asegurar la fiabilidad y robustez del sistema en diferentes escenarios.

3.2 Descripción del cultivo de mora a monitorear

Para el desarrollo del proyecto, se realizó una investigación bibliográfica y de campo, enfocada en la definición y documentación de cultivos de mora a nivel nacional y específicamente en la provincia de Tungurahua, en la cual existen diversos tipos de moras, pero la más cultivada es la mora colombiana (Rubus Glaucus Benth) por sus

diferentes características morfológicas, por lo cual ha sido seleccionada este tipo de mora para la investigación del proyecto.

El cultivo de mora se encuentra ubicado en la provincia de Tungurahua, cantón Ambato parroquia Montalvo, las coordenadas de la ubicación son las siguientes 1°18'54.3"S 78°36'38.8"W, además en la Figura 39 se muestra el lugar exacto de donde está ubicado el cultivo de mora.

Figura 39. Ubicación del cultivo de mora.

La variedad de mora cultivada es la colombiana con espinas, el cultivo de mora cuenta con 210 plantas, las cuales están ubicadas a una distancia de 2 metros entre plantas y 1.5 metros entre surcos como se muestra en Figura 40, lo que facilita el crecimiento adecuado de cada una de las plantas. Se lleva a cabo un manejo agronómico como la poda en donde se eliminan las ramas guía que crecen en desmedida y podrían opacar el desarrollo de las ramas productoras de flores y frutos. Asimismo, se podan las ramas inferiores más pequeñas, ya que su desarrollo limitado impide que produzcan una cantidad de frutos, lo que contribuye a una mejora la calidad de la cosecha.

Figura 40. Cultivo de mora desde tomada con el dron.

El cultivo de mora tiene aproximadamente tres años de crecimiento, estas plantas ya se encuentran en una fase productiva estable, permitiendo una buena cosecha de frutos. La infraestructura de riego implementada utiliza un sistema de goteo asistido por una bomba de agua, lo que garantiza una distribución precisa y controlada del recurso hídrico que optimiza el consumo de agua y promueve un desarrollo adecuado de las plantas, contribuyendo a la sostenibilidad del cultivo.

3.3 Desarrollo general de la propuesta

3.3.1 Selección de Tecnología

a. Selección de UAV

En la Selección de UAV, se realizó una comparación en la entre diferentes drones utilizados en la agricultura de precisión como se muestra en la Tabla 11. En este caso, se ha elegido el DJI Mini 2 como el más adecuado para el sistema de estimación de producción de mora utilizando visión artificial y planificación de trayectorias. Se ha seleccionado principalmente debido a sus capacidades avanzadas, como la estabilización de imágenes, su peso liviano, la posibilidad de ser programado para vuelos autónomos, y su alta precisión en la recolección de datos.

b. Selección de microcomputador

A continuación, en la Tabla 16, se presenta la comparación de las características más importantes de los microcomputadores investigados en el Capítulo I (Microcomputadores). Para el sistema de estimación de producción de mora empleando visión artificial y machine learning, se ha seleccionado la NVIDIA Jetson Nano. Esta elección es clave debido a su capacidad para ejecutar tareas complejas de inteligencia y visión artificiales directamente en campo, lo que es esencial para la eficiencia del sistema.

La Jetson Nano se destaca por su procesador ARM Cortex-A57 de cuatro núcleos y su GPU NVIDIA Maxwell con 128 núcleos CUDA, lo que proporciona la potencia de procesamiento necesaria para ejecutar algoritmos de aprendizaje profundo y análisis de imágenes en tiempo real. Además, su eficiencia energética es fundamental en

entornos agrícolas donde el acceso a fuentes de energía es limitado. Esto le permite operar de manera autónoma y realizar todo el procesamiento de datos localmente, sin depender de una conexión constante a la nube, lo cual es esencial en áreas rurales con conectividad a internet restringida.

c. Periféricos de entrada y salida

Para el sistema de estimación de producción de mora mediante visión artificial y machine learning, se emplean los siguientes periféricos de entrada y salida, que son esenciales para la interacción y el control del sistema durante el desarrollo y ejecución del proyecto:

- ***Monitor***

Es un periférico de salida fundamental que muestra visualmente la información procesada por el microcomputador. Este incluye las imágenes capturadas por el dron, los gráficos generados por los algoritmos de machine learning, y otros datos visuales clave para la interacción con el sistema. El monitor proporciona una interfaz visual que facilita el control del sistema y la verificación de resultados en tiempo real.

- ***Mouse***

Periférico de entrada que permite controlar el sistema de manera intuitiva mediante el desplazamiento del cursor en pantalla. El mouse es especialmente útil durante las fases de desarrollo del software, ya que facilita la navegación por los entornos de desarrollo (como Google Colab y PyCharm) y la interacción con las interfaces gráficas de usuario que se implementen para el monitoreo de la producción de mora.

- ***Teclado***

Dispositivo de entrada clave para la interacción con el sistema, permitiendo al usuario introducir comandos, escribir código y realizar configuraciones necesarias. Durante el desarrollo y ejecución de este sistema, el teclado es indispensable para modificar parámetros, entrenar modelos, y realizar ajustes en los algoritmos de machine learning implementados.

3.3.2 Selección de software

a. Lenguajes de Programación

Mediante la Tabla 5 descrita en el Capítulo I, la selección del lenguaje es Python debido a que ofrece un conjunto extenso de bibliotecas y frameworks especializados para la inteligencia artificial, aprendizaje profundo y procesamiento de imágenes, como OpenCV, TensorFlow, PyTorch y YOLOv8.

b. Entorno de desarrollo integrado

A través del análisis realizado en la Tabla 6, se identificaron Google Colab y PyCharm como las herramientas más adecuadas para el desarrollo del sistema de estimación de producción de mora utilizando visión artificial y machine learning. Google Colab se elige debido a que ofrece recursos gratuitos de GPU y TPU para llevar a cabo tareas de aprendizaje profundo, facilitando el procesamiento de imágenes en la nube. Además, permite el uso de bibliotecas y frameworks actualizados o antiguos compatibles con inteligencia artificial, así como el acceso directo a los datos de entrenamiento mediante Google Drive, optimizando los recursos de hardware tanto locales como en la nube. Por otro lado, PyCharm se selecciona para la implementación del código del modelo entrenado por su capacidad de crear entornos virtuales con las bibliotecas necesarias utilizadas en el proyecto. PyCharm también proporciona herramientas de depuración avanzadas y gestión eficiente de paquetes, lo que facilita el desarrollo del software para el procesamiento local de datos y la integración con dispositivos como drones y microordenadores.

c. Modelos de red neuronal

En la Tabla 15, descrita en el Capítulo I sobre los modelos preentrenados aplicados en la clasificación de cultivos, se analizan diferentes modelos de redes neuronales profundas. Entre ellos, YOLOv8m destacó como el modelo más adecuado para este proyecto debido a su alta precisión, velocidad de inferencia y capacidad para realizar detecciones en tiempo real.

Los modelos evaluados fueron YOLOv8m y VGG16, considerando sus características técnicas, funcionalidades principales y aplicaciones específicas en el ámbito agrícola. En este análisis, YOLOv8m demostró ser superior en precisión, especialmente para tareas de detección de objetos múltiples, como el conteo de frutos en plantas. Su desempeño sobresaliente en entornos dinámicos lo convierte en una herramienta ideal para aplicaciones de agricultura de precisión.

d. Base de datos

Mongo DB es la mejor opción si planeas trabajar con grandes cantidades de datos, utilizar análisis complejos o realizar operaciones avanzadas como manejo de datos proporciona una base sólida y flexible para el manejo de datos complejos, integraciones con herramientas de análisis y machine learning, y escalabilidad para futuros desarrollos como se muestra en la Tabla 18. Esto puede resultar en un sistema más eficiente y efectivo para la toma de decisiones en la producción agrícola.

3.3.3 Arquitectura del sistema

El sistema de estimación de producción de mora consta de varias etapas que permiten recopilar, procesar, analizar y visualizar la información obtenida del cultivo para realizar predicciones precisas. En la Figura 41 se muestra un esquema general de las etapas del sistema, que abarcan desde la adquisición de datos hasta la visualización de las estimaciones. A continuación, se describe cada una de estas etapas para proporcionar una mayor comprensión del funcionamiento completo del sistema.

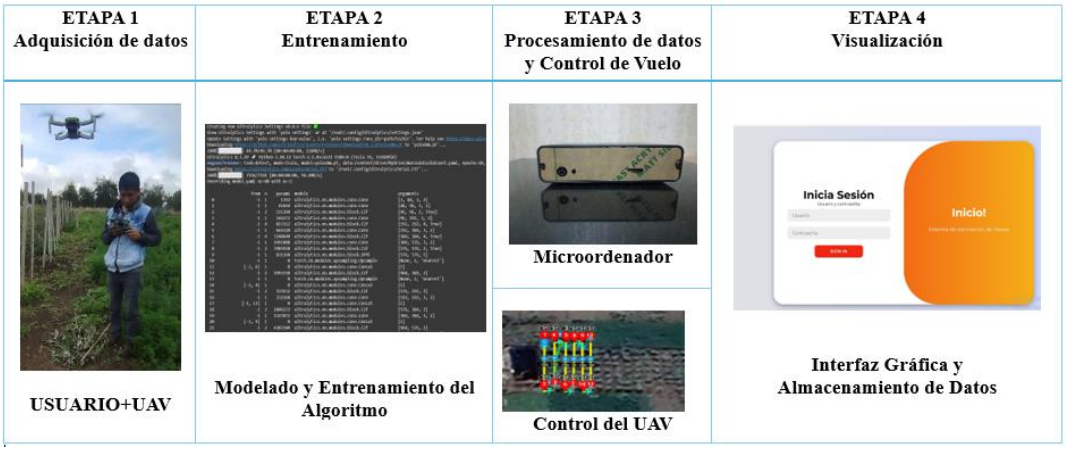

Figura 41. Etapas del Esquema general del sistema.

En la Figura 42 se muestra el esquema funcional más detallado del sistema que se va a implementar en el presente proyecto de investigación, con sus respectivas 4 etapas en las cuales se implementaran diversos recursos y materiales por cada etapa que se lo va a detallar posteriormente.

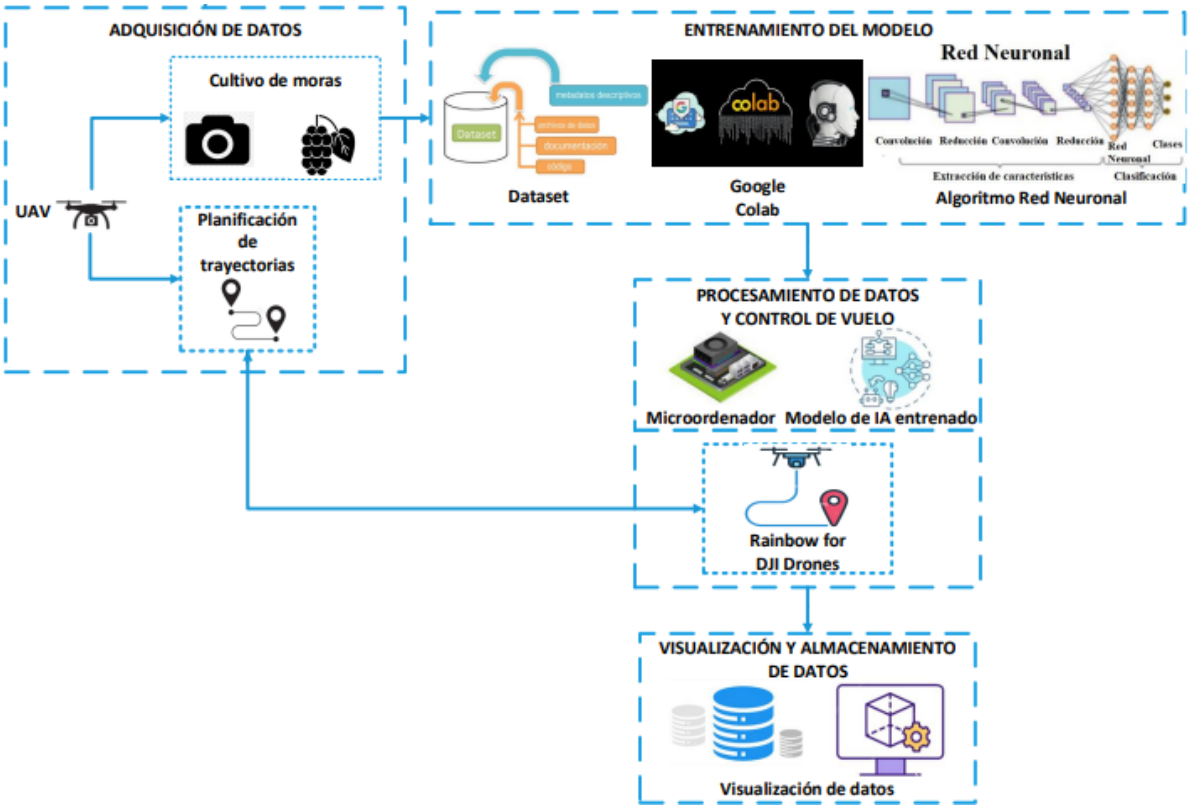

Figura 42. Esquema Funcional del sistema.

a. Etapa 1 Adquisición de datos

En la Figura 43 se muestra la etapa 1 donde se capturan imágenes y videos del cultivo de mora utilizando un dron (UAV) equipado con una cámara RGB. El dron sigue una trayectoria planificada para cubrir toda el área de cultivo y garantizar la adquisición completa de los datos visuales necesarios para la estimación.

También se pueden establecer puntos de control para que el dron tome imágenes en lugares específicos.

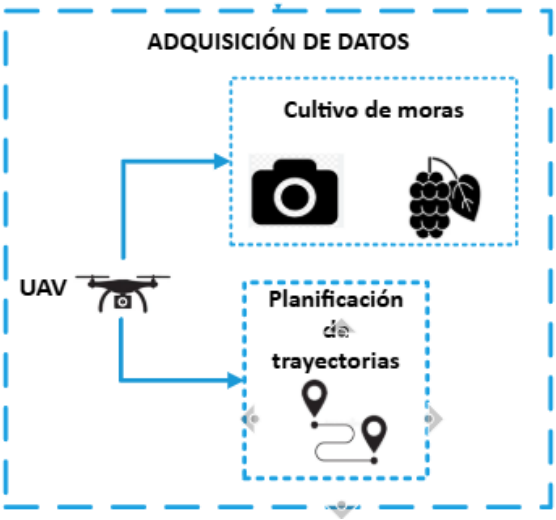

Figura 43. Etapa 1 adquisición de datos.

b. Etapa 2 Entrenamiento del Modelo

En la Figura 44 se muestra la etapa 2, en esta etapa se realiza el entrenamiento del modelo de detección y clasificación de moras utilizando algoritmos de machine learning y visión artificial. Se emplea YOLOv8 para detectar, contar y clasificar las moras por color. El entrenamiento se realiza sobre un conjunto de datos previamente etiquetados, lo que permite al modelo reconocer diferentes características relevantes como el estado de maduración de la mora.

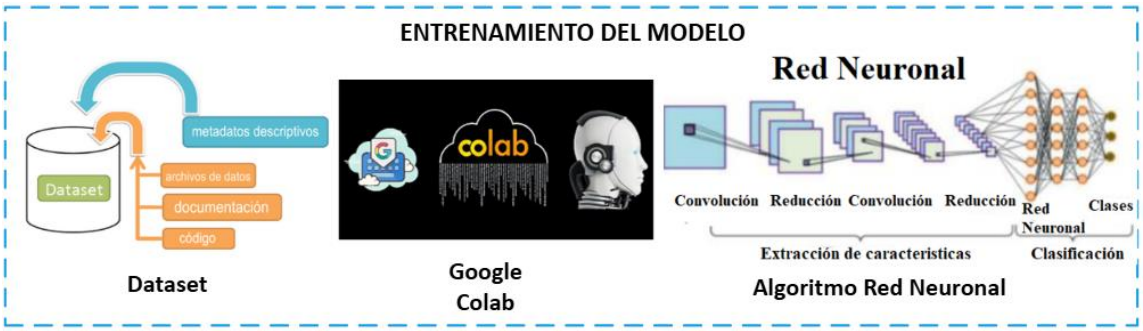

Figura 44. Etapa 2 entrenamiento del modelo.

- ***Capas del Algoritmo***

La arquitectura de la red neuronal convolucional aplicada en este sistema tiene como objetivo extraer y aprender características relevantes de imágenes capturadas en el proceso de estimación de producción de cultivo de mora. En la Figura 45 notamos que

el modelo YOLOv8 empleado para la detección de moras se organiza en tres secciones principales: entrada, capas ocultas y salida, las cuales trabajan de manera conjunta para realizar una clasificación efectiva de los frutos según su estado de maduración (verde, rojo y morado).

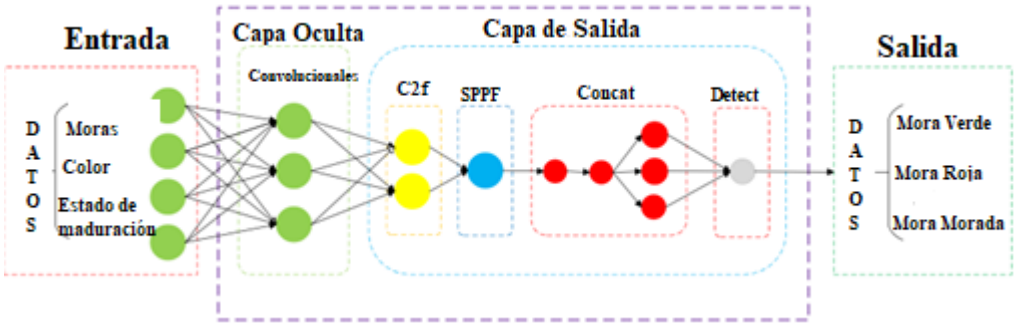

Figura 45. Capas de Algoritmo

Cada tipo de capa cumple una función esencial en el flujo de procesamiento:

Entrada: El sistema recibe los datos iniciales que comprenden las características visuales de las moras, como el color y el estado de maduración.

Capas ocultas: Dentro de esta sección, se desarrollan las transformaciones y aprendizajes esenciales para la detección y clasificación de los frutos:

- ✓ **Capas convolucionales:** Detectan y extraen atributos de las imágenes, como los bordes, las texturas y los patrones asociados a las etapas de maduración.
- ✓ **C2f:** Optimizan la extracción de características al reutilizar información entre las conexiones internas, reduciendo redundancias.
- ✓ **SPPF (Spatial Pyramid Pooling - Fast):** Realizan un resumen espacial de las características en diferentes escalas.
- ✓ **Concat:** Combina las características procesadas en las capas previas, integrando la información para aumentar la precisión del modelo.

Capas de salida: La última etapa incluye la capa de detección que clasifica las moras en tres categorías: verde, roja y morada. Esta capa final emplea la información

procesada en las secciones anteriores para identificar y contar los frutos según su estado de maduración.

Salida: El sistema genera los datos clasificados, mostrando los resultados en términos de cantidad de moras detectadas para cada estado de maduración, facilitando su interpretación y uso en el análisis de producción.

c. Etapa 3 Procesamiento de Datos y Control de vuelo

Una vez adquiridas las imágenes, estas se procesan en un microordenador (como NVIDIA Jetson Nano) para ejecutar los modelos de detección. Este procesamiento incluye:

- ✓ Preprocesamiento: Normalización, escalado y ajuste de las imágenes.
- ✓ Detección: Aplicación del modelo entrenado para identificar y contar las moras.
- ✓ Postprocesamiento: Filtrado de las detecciones, eliminación de duplicados, y organización de la información.
- ✓ Control de la trayectoria: El Rainbow for DJI se conecta al UAV para realizar las trayectorias en el cultivo para que este sea más eficiente y autónomo.

En la Figura 46 se muestra cómo se establece el procesamiento de los datos con la Jetson Nano y la planificación de trayectorias con la aplicación Rainbow for DJI Drones.

Figura 46. Etapa 3 de procesamiento de los datos y control de vuelo.

d. Etapa 4 Visualización y Almacenamiento

En la Figura 47 se muestra la etapa 4 donde los resultados del análisis se presentan en una interfaz gráfica para que el usuario pueda visualizar los datos de producción estimada, incluyendo estadísticas de conteo por color y tamaño. Además, se almacenan los datos obtenidos para un análisis histórico y toma de decisiones futuras sobre el cultivo.

Figura 47. Etapa 4 Visualización y almacenamiento de datos

e. Algoritmo general del sistema

A continuación, se proporciona una descripción detallada del algoritmo representado en el diagrama de flujo que se muestra en la Figura 48.

El algoritmo inicia el sistema y el entorno de trabajo. El dron vuela sobre el cultivo, siguiendo trayectorias planificadas y capturando el video. En caso de que las imágenes detectadas no sean validas, el sistema realizará una búsqueda adicional hasta encontrar uno. Si los videos capturados son válidos, se procesan utilizando algoritmos de visión artificial para detectar moras. Luego, el algoritmo procede a la clasificación del fruto, evaluando el porcentaje o nivel de madurez de acuerdo con el objeto detectado y dándole un id para no repetir el conteo. Posteriormente, utilizando un algoritmo, se estima la cantidad de producción basada en las características de las moras detectadas. En caso de que en cualquier punto del proceso no reconozca ningún objeto o la imagen no sea válida el sistema regresa al inicio para seguir realizando la detección y garantizar una monitorización continua del cultivo de mora.

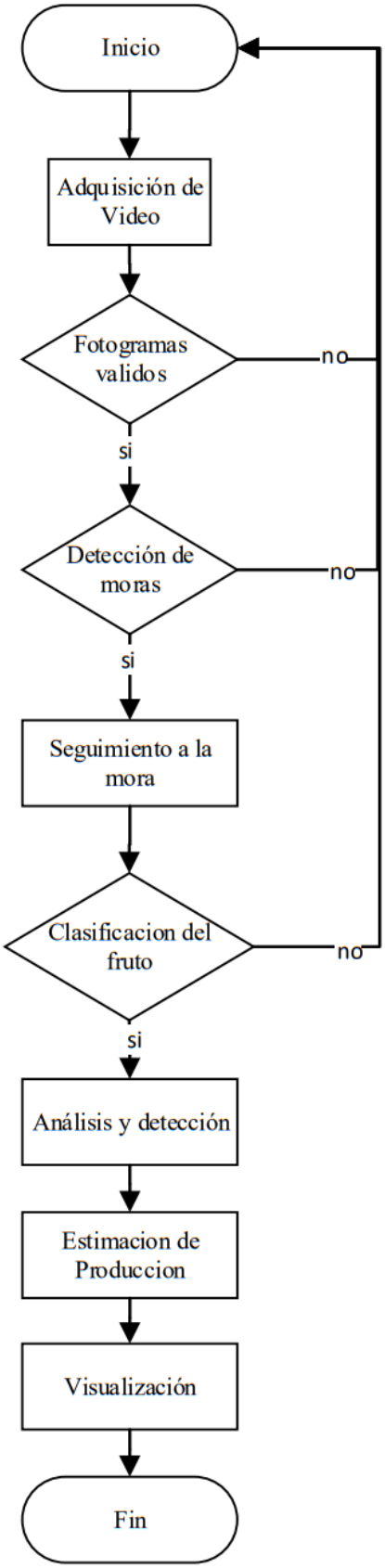

Figura 48. Diagrama de flujo general del algoritmo del sistema.

3.3.4 Diseño del sistema

a. Adquisición de Datos

- ***Posicionamiento Usuario-Sistema***

Para el sistema de adquisición de imágenes en el proyecto de estimación de producción de cultivo de mora, se requiere un posicionamiento óptimo entre la cámara 4K el área de cultivo. Al emplear un dron, la distancia y altura de vuelo deben ajustarse cuidadosamente para capturar imágenes nítidas y detalladas de las moras sin comprometer la calidad de enfoque y resolución. El objetivo es garantizar que la cámara capte con precisión las características individuales de cada planta y fruto, incluso en áreas amplias, para permitir un análisis efectivo en las etapas posteriores de procesamiento de imágenes y estimación de producción. La configuración del sistema se ejemplifica en la Figura 49, que ilustra la disposición de la cámara y el dron sobre la parcela de cultivo.

Figura 49. Escenario del sistema en el cultivo de mora.

- ***Control de vuelo autónomo***

En la Figura 50 se muestra el diagrama de flujo describe el funcionamiento de la planificación de trayectorias con el dron, el dron mediante la aplicación Rainbow for DJI Drones tiene modo de vuelo manual y el automático en la cual se planifica las trayectorias.

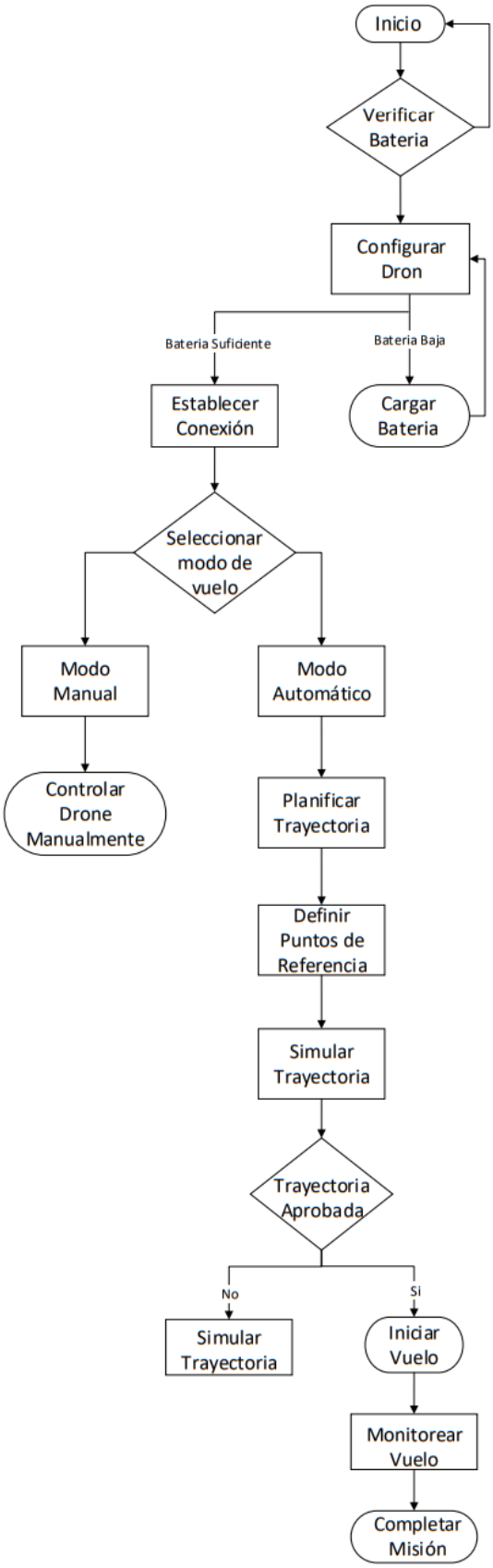

Figura 50. Diagrama de flujo del control de vuelo autónomo.

El flujo comienza con la verificación de la batería del dron asegurándose que tenga suficiente carga para continuar, sino se procede a cargarla antes de configurar el dron y establecer la conexión.

Después se selecciona el modo de vuelo: manual donde es controlado por el usuario o el automático donde se planifica la trayectoria.

En el modo automático por medio del GPS del dron nos ubicamos de manera exacta en el lugar donde vamos a poner los puntos de referencia o way points para la trayectoria, después despegamos el dron donde se comienza a monitorear por medio del celular y la aplicación para ver si cubre los puntos requeridos en la planificación y cumpla la misión, si no se aprueba se ajusta los parámetros de la trayectoria hasta tener una eficiencia en el proceso que cope los puntos necesarios.

Planificación de Trayectorias en el cultivo de mora

El terreno en donde se encuentra ubicado el cultivo de mora como se muestra en la Figura 40 tomada con el dron se establece un terreno plano con un amplio espacio entre surcos lo cual establece que no existen obstáculos importantes entre los surcos para que el dron realice las trayectorias.

Para la planificación de trayectorias se realizó mediante la aplicación Rainbow for DJI Drones que permite realizar misiones al dron mediante puntos de referencia la cual se conecta a satélites para una geolocalización del dron, por lo que cada fila es un punto de referencia en la cual el dron graba los cultivos de mora para reconocer los frutos.

Pasos para la planificación de trayectorias

El primer paso es conectar por medio de un cable OTC del celular o el monitor por el cual se va a ingresar a la interfaz de la aplicación Rainbown for DJI Drones hacia la palanca de control del dron, después se selecciona el tipo de dron que se va a utilizar para luego realizar la conexión, en este caso el dron DJI mini 2 como se muestra en la Figura 51.

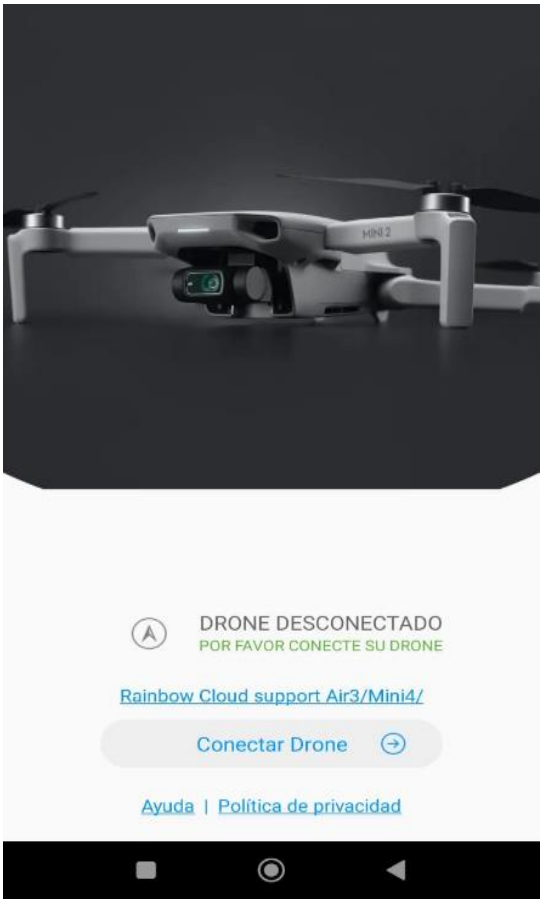

Figura 51. Conexión del dron con la aplicación.

Después aparece la siguiente interfaz de usuario (UI) como se muestra en la Figura 52 en la cual están diversos parámetros importantes de la UI.

Figura 52. Interfaz de Usuario de la aplicación Rainbow for DJI Drones.

Los parámetros más importantes que se visualiza en la Figura 52 son los siguientes:

1. Estado de conexión del dispositivo

2. Modo avión o dron

3. Numero de satélites

4. Cantidad eléctrica de control remoto

5. Cantidad eléctrica del dron

6. Voltaje

7. Señal del control remoto

8. Señal del dron

9. Configuración general

10. Ajuste de la cámara

11. Cambio de modo de vuelo

12. Brújula

12.1 Altitud y distancia de vuelo

12.2 Velocidad horizontal y vertical

13. Ventana de descripción general

14. Altura del sistema virtual

En la opción 11 en el cambio de modo de vuelo existe 6 tipos, en este caso se utilizó el way point el cual ayudó a tomar puntos de referencia para la planificación de trayectorias según las condiciones y estado del terreno en la cual se encuentra el cultivo de mora, que anteriormente se mencionó que no hay obstáculos entre surcos tal como se muestra la Figura 53:

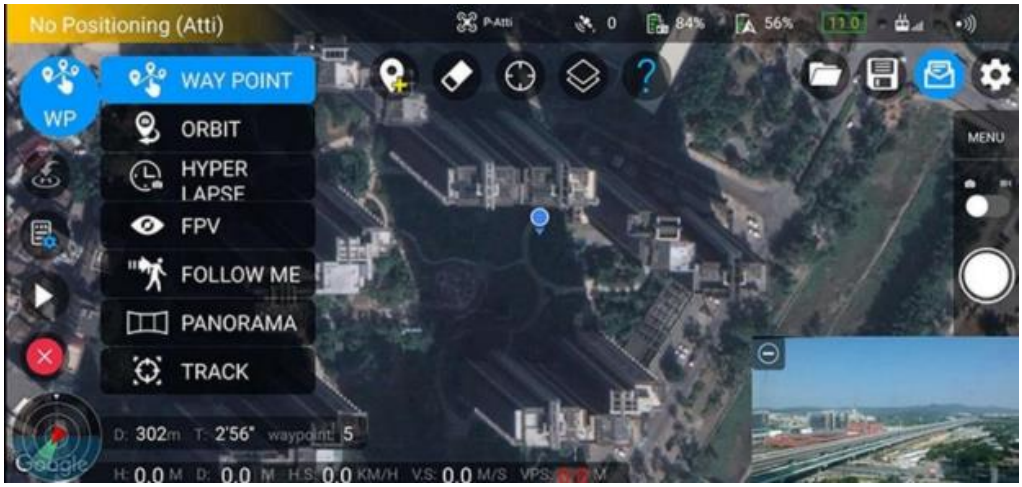

Figura 53. Modos de vuelo del dron.

A continuación, en la Figura 54 se muestra los parámetros del modo de vuelo en way point importantes para el manejo de la aplicación y observar los datos de vuelo del dron:

Figura 54. Parámetros importantes en modo Way point.

1. Archivo Kmz.
2. Trayectoria visualización
3. Puntos de referencia
4. Borrador de puntos de referencia
5. Regreso hacia el centro de pantalla
6. Cambio de modo de capa del mapa

7. Way points establecidos

8. Archivos locales

9. Guardar archivos

10. Retorno del punto

11. Establecer propiedad de la misión

12. Inicio de misión

13. Terminar misión

14. Distancia de vuelo recorrido

15. Tiempo de vuelo

16. Cantidad de puntos de referencia

Para empezar a editar la misión o la trayectoria se edita la misión utilizando los puntos de referencia o el punto “3” mostrado en la Figura 54 en donde se requiera empezar la trayectoria se desplegará 3 opciones la cual se escoge la primera como se muestra en Figura 55:

Figura 55. Selección del primer icono para los puntos de referencia.

Después se añade varios puntos en la interfaz de vuelo donde se encuentra el mapa mostrando el cultivo de mora, en este caso se coloca el primer way point en el inicio del cultivo y de manera ordenada se colocan las demás por cada surco para que el dron haga un barrido de cada parcela como se muestra en la Figura 56:

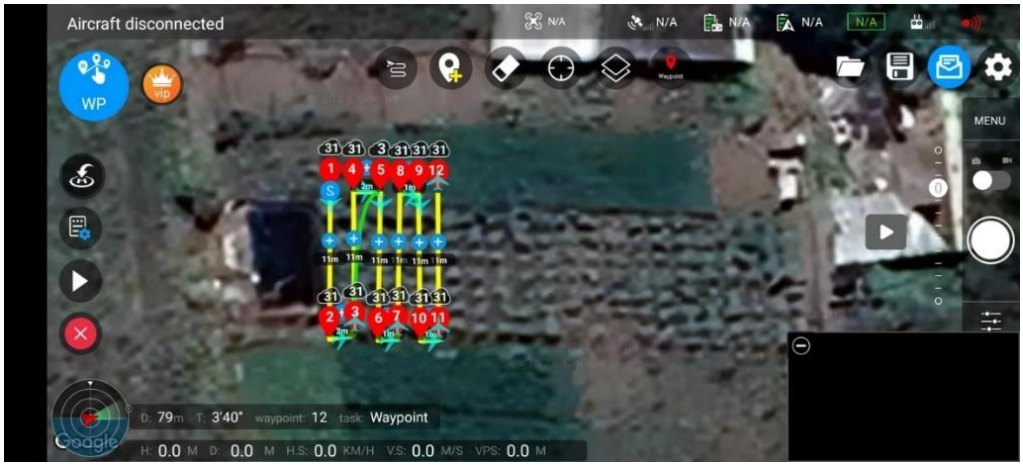

Figura 56. Puntos de referencia establecidos para copar toda el área del cultivo de mora.

Después debemos establecer la altitud del vuelo del dron y la velocidad en la cual se moverá el dron, la distancia y la rapidez de vuelo se establece anteriormente ya que se debe tener presente que el dron debe capturar por video la mayor cantidad de moras posibles por la altura el por la velocidad que pase por cada planta de mora para que el sistema sea la más eficaz posible. Por ello se estableció que la altura en la cual se maneje el dron sea de 2 metros a una velocidad de 0.2 m/s.

Para establecer la altura y la velocidad de cada punto se debe seleccionar cada punto o way point ya establecido se despliega una ventana en la cual se puede modificar como se muestra en la Figura 57 y la Figura 58.

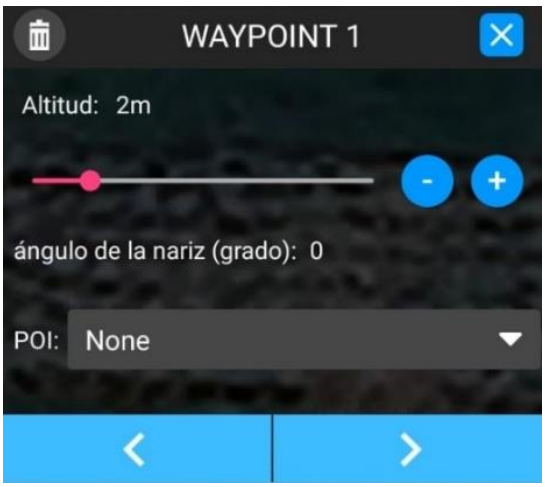

Figura 57. Altura establecida para el dron.

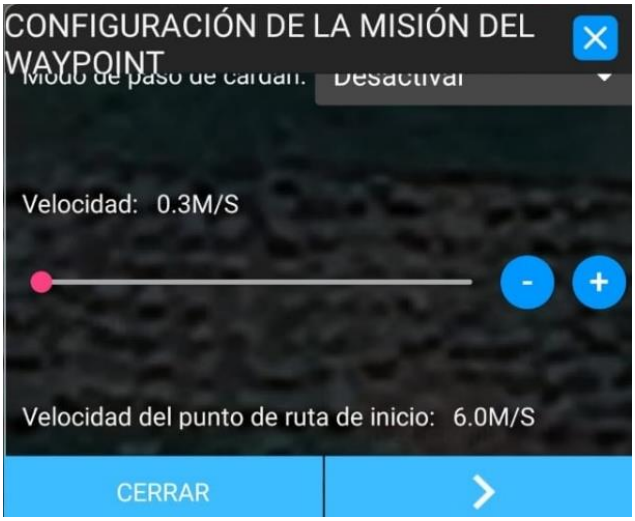

Figura 58. Velocidad establecida para el dron.

- ***Adquisición de video***

En la Figura 59 se ilustra el flujo de este proceso, desde el inicio del vuelo del dron hasta la captura y almacenamiento de los frames, en un sistema de almacenamiento local.

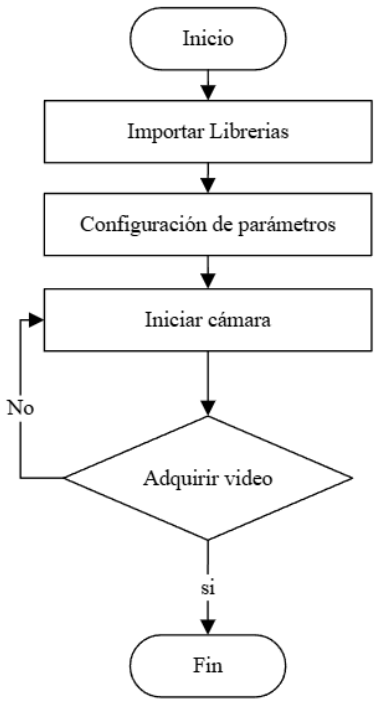

Figura 59. Diagrama de flujo para la adquisición de video.

La captura de video se realiza utilizando una cámara equipada en el dron DJI Mini 2. El proceso de obtención de datos comienza con la planificación y ejecución de las trayectorias de vuelo sobre el cultivo de mora. La cámara permite capturar imágenes de alta resolución, lo que asegura una alta precisión en la adquisición de detalles necesarios para el procesamiento.

Bibliotecas requeridas

Para la adquisición y procesamiento de imágenes las siguientes librerías son necesarias para el manejo del servidor web, procesamiento de imágenes, detección con Yolo, seguimiento con SORT, almacenamiento en MongoDB, simulación de Monte Carlo y manejo de fechas:

- ✓ openCV
- ✓ pyTorch
- ✓ NumPy
- ✓ flask
- ✓ Ultralytics
- ✓ Sort
- ✓ date

En esta sección se importan las bibliotecas necesarias para la ejecución del script:

- ✓ ***Flask:*** Biblioteca para desarrollo de aplicaciónes web que permite interactuar con el sistema de manera intuitiva.

- ✓ ***OpenCV (cv2):*** Para capturar fotogramas de video de la cámara del dron y realizar tareas de procesamiento de imágenes como el etiquetado de las moras.

- ✓ ***NumPy:*** Librería para el manejo de datos de las detecciones como arrays y para interactuar con otras librerías como OpenCV y SORT.

- ✓ ***Torch:*** Herramienta empleada para implementar el modelo YOLOv8, enfocado en la detección de los objetos.

- ✓ ***SORT:*** Algoritmo de rastreo para las moras detectadas en los fotogramas el cual asigna identificadores únicos (ID de seguimiento).

- ✓ ***Pymongo:*** Biblioteca que facilita la conexión y gestión de la base de datos MongoDB, donde se almacenan los resultados procesados.

Importar las librerías

Revisar el código completo en ANEXO F.

El siguiente código corresponde a un script para la detección y conteo de moras utilizando YOLOv8, OpenCV, SORT y Flask para la integración con una interfaz web. Este algoritmo es parte fundamental del sistema automatizado de estimación de producción. A continuación, se describe la estructura inicial del código:

```
# Importación de Flask y funciones auxiliares para manejar respuestas HTTP
from flask import Flask, Response, send_file, jsonify, render_template, request
import cv2  # OpenCV para manipulación de imágenes y video
import os  # Módulo para interactuar con el sistema operativo
import numpy as np  # Biblioteca para cálculos numéricos
from ultralytics import YOLO  # Importación del modelo YOLO para detección de objetos
from sort import Sort  # Importación del algoritmo de seguimiento de objetos Sort
import pymongo  # Biblioteca para interactuar con bases de datos MongoDB
```

Configurar los parámetros: Se establece la conexión con la base de datos, inicializa variables y define los modelos de detección y seguimiento.

```
# Inicialización de la aplicación Flask
app = Flask(__name__, template_folder=os.path.dirname(__file__))

# Configuración de la conexión a MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017/")  # Conexión al servidor MongoDB local
db = client["detecciones_db"]  # Selección de la base de datos
collection = db["detecciones2"]  # Selección de la colección donde se almacenan las detecciones

# Inicialización de variables globales
camera = None  # Variable para almacenar la cámara
video_running = False  # Bandera para controlar el estado del video
txt-clase_0_ids = set()  # Conjunto para almacenar IDs de objetos de clase 0
txt-clase_1_ids = set()  # Conjunto para almacenar IDs de objetos de clase 1
txt-clase_2_ids = set()  # Conjunto para almacenar IDs de objetos de clase 2
```

Modelo YOLO y rastreo: Se configura el modelo de detección con el modelo preentrenado.

```
# Inicialización del modelo YOLO y los rastreadores
model = YOLO("best1_2.pt")  # Carga del modelo YOLO entrenado
tracker = Sort()  # Inicialización del rastreador para la clase 0
tracker2 = Sort()  # Inicialización del rastreador para la clase 1
tracker3 = Sort()  # Inicialización del rastreador para la clase 2
```

El modelo YOLO previamente entrenado se carga para realizar detecciones en las imágenes transmitidas por el dron. Se inicializan tres instancias del algoritmo SORT para realizar el seguimiento de las detecciones (rojas, verdes y moradas).

Inicialización de la cámara: Configura la cámara del dron para capturar datos en tiempo real. La dirección RTMP se utiliza para transmitir el video del dron al sistema.

```
camera = cv2.VideoCapture("rtmp://192.168.100.13:1935/live/test")
```

Función para la generación del video: Esta función inicia el flujo de video desde el dron y valida que la cámara se haya abierto correctamente.

```
@app.route('/start_video', methods=['POST'])
def start_video():
    global camera, video_running
    if not video_running:
        camera = cv2.VideoCapture("rtmp://192.168.100.13:1935/live/test")
        if not camera.isOpened():
            return jsonify({"status": "Error al abrir la cámara"}), 500
        video_running = True
    return jsonify({"status": "Video iniciado"})
```

b. Procesamiento de datos

Después de capturar la escena, los fotogramas se almacenan por defecto en formato RGB. Posteriormente, en la Figura 60 observamos que el procesamiento que incluye las siguientes etapas:

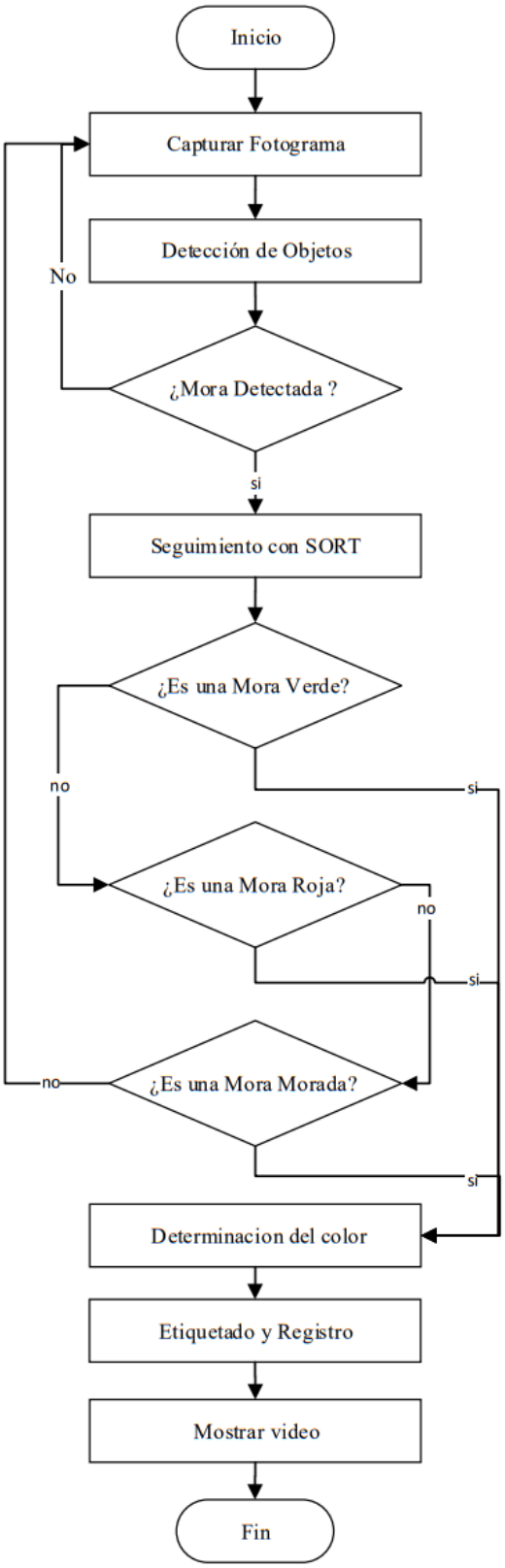

Figura 60. Diagrama de flujo para el procesamiento de datos

Capturar Fotograma

Inicialización del flujo de video: El flujo comienza obteniendo cada cuadro de la transmisión de video. Se procesa el flujo de video frame por frame. Cada frame es analizado por el modelo YOLO, que identifica y clasifica las moras según su color. Los objetos detectados son seguidos mediante el algoritmo SORT, y se actualizan las listas correspondientes (clase_0_ids, clase_1_ids, clase_2_ids).

```
# Generador de frames para el flujo de video
def gen_frames():
    global camera, video_running
    while video_running:
        success, frame = camera.read()  # Captura un frame del video
        if not success:
            break
        else:
            # Detección de objetos usando YOLO
            results = model(frame, stream=True, classes=[0, 1, 2])
```

- ✓ camera.read(): Captura un cuadro del video.
- ✓ Si no hay éxito en la captura (por ejemplo, si se llega al final del video), el bucle se detiene.
- ✓ Si el cuadro es capturado correctamente, pasa al seguimiento.

Procesamiento en tiempo real para identificación y clasificación

Esta sección del código se encarga de realizar las operaciones principales de procesamiento en tiempo real: detección de objetos, clasificación por tipos de mora (verdes, rojas y moradas) y seguimiento de las detecciones con rastreadores. A continuación, se divide en partes para facilitar su entendimiento.

Detección de objetos con YOLOv8: Detectar objetos en el fotograma (en este caso, moras). Este modelo utiliza una confianza mínima de detección (conf=0.7).

```
results = model(frame, conf=0.7, show=False, show_boxes=False,
                show_labels=False, stream=True, classes=[0, 1])
```

Identifica las ROIs (Regions of Interest) donde se encuentran las moras en el fotograma. Luego filtra detecciones únicamente para las clases especificadas:

- ✓ Clase 0: Moras rojas.
- ✓ Clase 1: Moras Verdes.
- ✓ Clase 2: Moras moradas.

Seguimiento de objetos con SORT: Las coordenadas de las detecciones se envían al rastreador SORT para asociarles un track_id único.

```
# Actualizar rastreadores
tracks = tracker.update(boxes)
tracks2 = tracker2.update(boxes2)
tracks3 = tracker3.update(boxes3)
```

- ✓ Cada conjunto de coordenadas (boxes, boxes2, boxes3) se actualiza con su respectivo rastreador SORT.
- ✓ Los rastreadores garantizan que cada objeto detectado mantenga su identidad a través de los cuadros.

Extracción de la región de interés (ROI): Extrae la región de la imagen correspondiente a la detección. Proporcionar los datos necesarios para determinar el color de la mora.

Determinación del Color del Objeto

Validación inicial: Si la región de la mora (region_mora) es nula o tiene tamaño cero, se devuelve 'otro'.

```
if region_mora is None or region_mora.size == 0:
    return 'otro'
```

Conversión a HSV (Hue, Saturation, Value): Las imágenes adquiridas son convertidas al espacio de color HSV para facilitar la detección de características específicas, como el color de las moras. Esto se realiza mediante la función

cv2.cvtColor de OpenCV, que permite cambiar el espacio de color según sea necesario. En la Figura 61 observamos la conversión de RGB a HSV.

```
hsv_region = cv2.cvtColor(region_mora, cv2.COLOR_BGR2HSV)
```

Figura 61. Imagen original RGB frente a imagen convertida a HSV

Definición de rangos HSV: Se definen rangos para los colores rojo, verde, y morado.

```
lower_green = np.array([40, 50, 50])   # Verde (tono bajo)
upper_green = np.array([80, 255, 255]) # Verde (tono alto)

lower_red = np.array([0, 100, 100])   # Rojo (tono bajo)
upper_red = np.array([180, 255, 255]) # Rojo (tono alto)

lower_purple = np.array([100, 50, 0])    # Morado (tono bajo)
upper_purple = np.array([160, 255, 255]) # Morado (tono alto)
```

Máscaras de color: Se aplican máscaras para aislar los píxeles de cada color. Filtrar la imagen para obtener solo la mora de color específico.

```
mask_red = cv2.inRange(hsv_region, lower_red, upper_red)
mask_green = cv2.inRange(hsv_region, lower_green, upper_green)
mask_purple = cv2.inRange(hsv_region, lower_purple, upper_purple)
```

Cálculo del promedio: Se calcula el color promedio en cada máscara para identificar el color dominante.

```
avg_red = np.mean(cv2.bitwise_and(region_mora, region_mora,
                                  mask=mask_red), axis=(0, 1))
```

Etiquetado y Registro

Dibujado de resultados en el cuadro: Se dibujan las cajas y etiquetas de las detecciones en el cuadro procesado.

```
# Dibujar rectángulos y etiquetas
for (xmin, ymin, xmax, ymax, track_id) in tracks:
    xmin, ymin, xmax, ymax = map(int, [xmin, ymin, xmax, ymax])
    cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 0, 255), 3)
    cv2.putText(frame, f"Mora {track_id} - Roja", (xmin, ymin - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 4)
    clase_0_ids.add(track_id)
```

- ✓ Cajas delimitadoras: Se dibuja un rectángulo sobre cada detección.
- ✓ Etiquetas: Se muestra información como el track_id y el color detectado de la mora.

Colores por clase:

- ✓ **Moras rojas:** (0, 0, 255).
- ✓ **Moras verdes:** (277, 131, 277).
- ✓ **Moras moradas:** (255, 0, 255).

Registro y visualización de resultados: Dibujar un rectángulo y un texto sobre cada detección en el fotograma.

```
cv2.rectangle(frame, (xmin3, ymin3), (xmax3, ymax3), color3, 3)
cv2.putText(frame, label_text3,
(xmin3, ymin3 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, color3, 4)
```

Función de almacenamiento de detecciones: Esta función calcula y almacena los resultados en la base de datos.

```python
# Función para almacenar detecciones en MongoDB
def guardar_detecciones():
    cont_rojas = len(clase_0_ids)  # Conteo de objetos clase 0
    cont_verdes = len(clase_1_ids)  # Conteo de objetos clase 1
    cont_moradas = len(clase_2_ids)  # Conteo de objetos clase 2
    totales = cont_rojas + cont_verdes + cont_moradas  # Suma total de objetos detectados

    # Creación del documento a almacenar en MongoDB
    documento = {
        "cantidad_rojas": cont_rojas,
        "cantidad_verdes": cont_verdes,
        "cantidad_moradas": cont_moradas,
        "totales": totales
    }
    collection.insert_one(documento)  # Inserción del documento en la base de datos
    print(f"Guardado: {documento}")  # Impresión del documento guardado
```

✓ Calcula el conteo total de moras detectadas por clase (rojas, verdes, moradas y brotes).

✓ Almacena los resultados en MongoDB para futuras consultas.

Mostrar Video

Conversión a formato transmisible

El cuadro procesado se convierte a formato JPEG y se prepara para su transmisión.

```python
# Codificar la imagen en formato JPEG
_, buffer = cv2.imencode('.jpg', frame)
yield (b'--frame\r\n'
       b'Content-Type: image/jpeg\r\n\r\n' + buffer.tobytes() + b'\r\n')
```

✓ cv2.imencode('.jpg', frame): Convierte el cuadro procesado en un archivo JPEG.

✓ yield: Envía el cuadro procesado al cliente en un flujo continuo de video.

Inicialización de la transmisión de video

Estas funciones manejan el inicio, detención y transmisión del video en tiempo real.

Inicio del video:

```python
# Ruta para iniciar el video
@app.route('/start_video', methods=['POST'])
def start_video():
    global camera, video_running
    if not video_running:
        camera = cv2.VideoCapture("VID_20241211_180521.mp4")
        if not camera.isOpened():
            return jsonify({"status": "Error al abrir la cámara"}), 500
        video_running = True
    return jsonify({"status": "Video iniciado"})
```

Detención del video:

```python
# Ruta para detener el video
@app.route('/stop_video', methods=['POST'])
def stop_video():
    global camera, video_running
    video_running = False
    guardar_detecciones()
    if camera:
        camera.release()
        camera = None
    return jsonify({"status": "Video detenido"})
```

Flujo de video:

```python
# Ruta para enviar el flujo de video
@app.route('/video_feed')
def video_feed():
    if not video_running or not camera:
        return Response("Video no activo", status=503)
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

c. Clasificación del Dataset para el entrenamiento

Los datos utilizados para el desarrollo del dataset se dividen en 3 categorías o estados de maduración del fruto de mora colombiana. En total se recopilaron 3000 imágenes empleadas tanto para la validación como para el entrenamiento del modelo. Del total de las 3000 imágenes, 1000 son de imágenes de moras verdes, 1000 de imágenes de moras rojas y por último 1000 imágenes de moras maduras o moradas.

Las imágenes se obtuvieron utilizando la cámara de alta gama, capturando información desde distintos ángulos y en diversas etapas de maduración del fruto, con el objetivo de garantizar la diversidad y precisión en la clasificación de las categorías mencionadas.

- ***Etiquetado de datos***

El etiquetado de datos se realizó mediante Labelme, un software compatible con el modelo de Yolo. Labelme permite seleccionar el área de interés en la imagen y etiquetar dependiendo a la clase que se desee asignar, de esta manera se identifica el estado de maduración de la mora obteniendo de esta manera las características esenciales en las imágenes necesarias en el entrenamiento. Este proceso de etiquetado genera un archivo necesario para que el modelo de entrenamiento encuentre los puntos de referencia en la imagen y a la clase que pertenece. En la Figura 62, se muestra el etiquetado de las moras con Labelme. Para el proceso de etiquetado se seleccionó el directorio en donde se encuentran los datos y el directorio donde se van a guardar los nuevos archivos generados. Se elije la opción de crear recuadro y se marca sobre la imagen el área de interés, finalmente se asigna la clase a la que pertenece y se genera el nuevo archivo con extensión json.

Figura 62. Anotación de datos utilizando Labelme.

- ***Estructura del dataset***

Para la creación del sistema que ayude a evaluar el estado de maduración de la mora y su respectiva clasificación se creó un dataset desde cero con ayuda de los datos que fueron adquiridos a través de la cámara, logrando recolectar información desde distintos ángulos y en distintas etapas de maduración de la mora. El dataset de se

compone de un total de 3000 imágenes distribuidas equitativamente entre las distintas etapas de maduración de la fruta: 1000 imágenes de moras verdes, 1000 de moras rojas y 1000 de moras moradas o maduras.

Para crear un sistema que evalúe los estados de maduración de la mora, se seleccionó una base de datos con muchas imágenes de diferentes estados de maduración de las moras. Estas imágenes fueron elegidas para que el sistema aprendiera a reconocer las características que indican los estados de maduración de la mora por medio del rango de color del fruto. La estructura del dataset fue diseñada para facilitar el entrenamiento del sistema, como se muestra en la Figura 63, el proceso de entrenamiento utilizó el 80% de las imágenes para el entrenamiento del modelo, mientras que el 20% restante se destinó a la validación, asegurando así un balance adecuado para evaluar el desempeño del sistema durante su desarrollo.

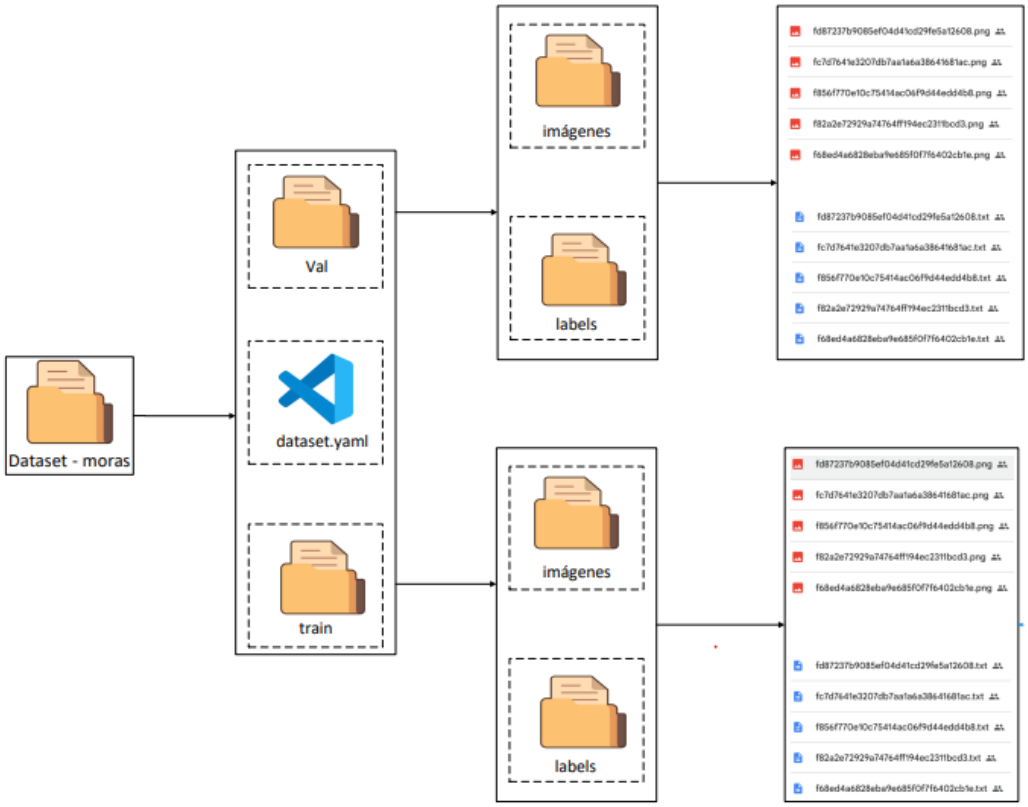

Figura 63. Arquitectura del dataset

La arquitectura de la base de datos está diseñada para facilitar el entrenamiento de modelos de aprendizaje automático y comprende dos componentes principales estructurados de forma específica. El primer componente, denominado "train", se

desglosa en dos subdirectorios: "images" y "labels". En el subdirectorio "images", se almacenan representaciones visuales de entidades de interés, que sirven como entrada para el entrenamiento del modelo. Por otro lado, el subdirectorio "labels" incluye metadatos anotados, principalmente las coordenadas de las cajas delimitadoras, que son generadas mediante el uso de la herramienta de anotación Labelme, proporcionando así las etiquetas de verdad de campo necesarias para el aprendizaje supervisado. El segundo componente es el directorio "val", que replica la estructura de "train" con subdirectorios "images" y "labels". Sin embargo, a diferencia del directorio de entrenamiento, las imágenes en "val" son exclusivamente para validación y no se exponen al modelo durante la fase de entrenamiento.

Finalmente se obtiene un archivo dataset.yaml en el que contiene la ruta de los archivos tanto de las etiquetas como el de las imágenes, el número de clases y el nombre de cada una de ellas. En la Figura 64, se muestra la estructura del archivo dataset.yaml.

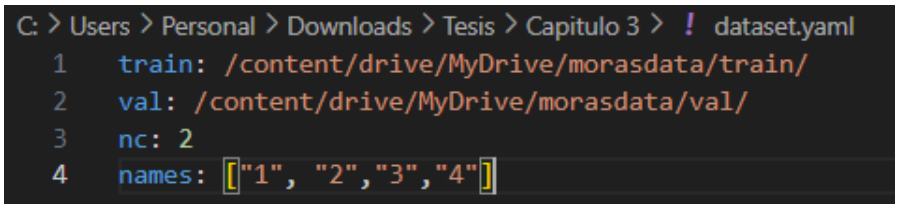

Figura 64. Archivo dataset.yaml

d. *Entrenamiento del modelo Yolo*

En esta sección se detalla el proceso de entrenamiento del modelo YOLOv8m, el cual ha sido seleccionado debido a su balance entre precisión y eficiencia computacional para el sistema de estimación de producción de mora. El modelo fue entrenado utilizando Google Colab para aprovechar su capacidad de GPUs gratuitas. En la Figura 65 observamos el entorno de desarrollo. Los pasos son los siguientes:

```
from google.colab import drive
drive.mount('/content/drive')
```

Clonar el repositorio YOLOv8 e instalar dependencias:

```
!git clone https://github.com/ultralytics/ultralytics.git
!pip install -r ultralytics/requirements.txt
```

- ***Código de Entrenamiento***

```
import os
from ultralytics import YOLO
# Definir el path al dataset
dataset_path = '/content/drive/MyDrive/MorasData/dataset.yaml'
# Verificar que la ruta del dataset existe
if not os.path.exists(dataset_path):
    raise FileNotFoundError(f"Dataset path '{dataset_path}' does not exist.")
# Cargar modelo preentrenado YOLOv8m
model = YOLO('yolov8m.pt')
# Entrenar el modelo
results = model.train(
    data=dataset_path,  # Ruta del dataset.yaml
    epochs=50,          # Número de épocas
    imgsz=640,          # Tamaño de las imágenes
    batch=-1,           # Tamaño de lote automático
    device=0            # Usar GPU
)
```

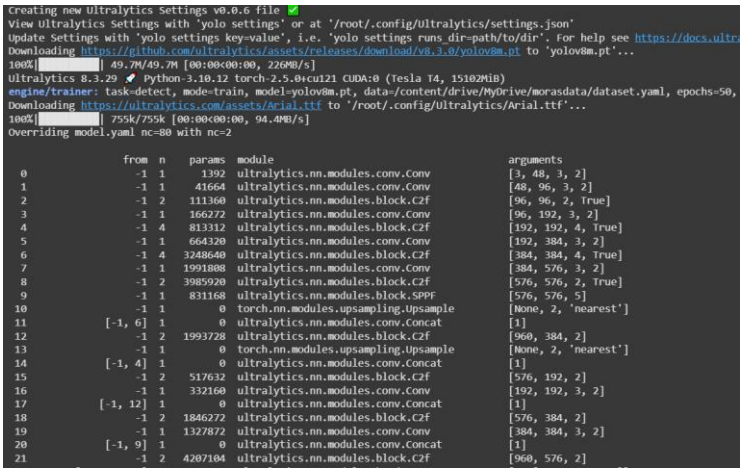

Figura 65. Cargando el modelo Yolov8m

En la Figura 66 se observa el Entrenamiento:

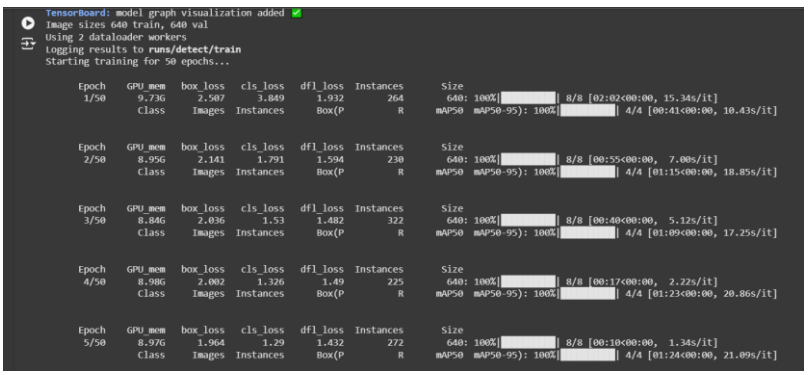

Figura 66. Entrenando el modelo por 50 Épocas

e. Entrenamiento modelo VGG16

Para el entrenamiento de este modelo se utilizó datos que se encuentran clasificados entre los distintos estados de maduración de mora (verde, roja y madura). Para este modelo se debe clasificar los datos de entrenamiento y validación en carpetas distintas, en cada una de ella se encuentra clasificado por el estado de maduración de la mora por carpetas que serían las clases que el modelo va a detectar, a diferencia del modelo Yolo este no emplea un etiquetado de sus datos facilitando el entrenamiento del modelo.

En la Figura 67, se muestra las librerías necesarias para el entrenamiento en las que destaca las librerías de numpy que es importante para el traqueo junto con la librería sort, además de PyTorch permite ajustar los parámetros del modelo para mejorar los resultados.

```
import time
# import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
# from torchvision import datasets
from torchvision import transforms
from torch.utils.data import DataLoader
from PIL import Image
```

Figura 67. Librerías para el modelo VGG16.

Los parámetros para el entrenamiento se definen en la Figura 68; donde se define el tamaño del lote, las épocas del entrenamiento, la tasa de aprendizaje y la redimensión de las imágenes a 32x32 pixeles.

```
# hyperparameter
# random_seed = 1
learning_rate = 0.00005
num_epochs = 50
batch_size = 10

num_classes = 4
IMG_SIZE = (32, 32)   # resize image
```

Figura 68. Parámetros del entrenamiento.

En la Figura 69 se define los directorios donde se encuentra los datos de entrenamiento y validación almacenados en Google Drive.

```
train_data_dir = '/content/drive/MyDrive/datos VGG16/entrenamiento'
validation_data_dir = '/content/drive/MyDrive/datos VGG16/validacion'
```

Figura 69. Instanciación de directorios de los datos.

Antes de empezar con el enteramiento se debe realizar un procesamiento de las imágenes para que cumplan con los parámetros definidos en la Figura 68. En la Figura 70, se muestra el código que permite modificar estos parámetros en cada uno de los datos. El modo de entrenamiento para el modelo en este caso es de categorías como se va a obtener varias salidas de la red neuronal entrenada, VGG16 utiliza 16 capas de entrenamiento.

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(width_shape, height_shape),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = valid_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(width_shape, height_shape),
    batch_size=batch_size,
    class_mode='categorical')
```

Figura 70. Codificación para ajustar los datos a las necesidades del modelo.

En la Tabla 22, se describe el modelo VGG16, a continuación, se detalla los parámetros de configuración del modelo:

Tabla 22. Parámetros de configuración del modelo.

Parámetro	Valor
Dispositivo (DEVICE)	cuda:0 o cpu
Tasa de aprendizaje	0.00005
Épocas (num_epochs)	100
Tamaño del lote (batch_size)	10
Número de clases (num_classes)	3
Tamaño de imagen (IMG_SIZE)	(32, 32)
Transformaciones	Resize, ToTensor
Función de pérdida	CrossEntropyLoss
Optimizador	Adam

f. Resultados de entrenamiento

- ***Resultados del modelo Yolo***

Al finalizar el entrenamiento, se obtuvieron los siguientes resultados que se muestran en la Tabla 23:

Tabla 23. Resultados del Entrenamiento.

Clase	Imágenes	Instancias	Box(P)	R	mAP50	mAP50-95
Todas	3930	3397	0.912	0.861	0.933	0.594
1	1530	1390	0.911	0.917	0.950	0.639
2	1280	1007	0.912	0.805	0.916	0.550
3	1120	1000	0.912	0.853	0.921	0.553

- ***Significado de las métricas***

Box(P): Indica el nivel de precisión alcanzado en la predicción de las cajas delimitadoras (bounding boxes).

R (Recall): Representa la sensibilidad del modelo, es decir, su capacidad para identificar todas las instancias presentes en el conjunto de datos.

mAP50: Promedio de precisión calculado para un umbral de IoU (Intersección sobre Unión) del 50%.

mAP50-95: Media de precisión promedio considerando umbrales IoU entre el 50% y el 95%.

Métricas Obtenidas

En la Tabla 24Tabla 24 se muestran las métricas obtenidas tras el entrenamiento de 500 épocas son las siguientes:

Tabla 24. Métricas del Entrenamiento

Métrica	Valor
Precisión (Precision)	0.95
Cobertura (Recall)	0.92
mAP50	0.94
Pérdida Total (Loss)	0.18

El modelo alcanzó un mAP50 del 93.3%, lo que indica una alta precisión en la detección de ambas clases (mora madura e inmadura). La precisión y el recall indican que el modelo es fiable tanto en identificar moras como en evitar falsas detecciones. Las detenciones realizadas por el modelo muestran una correcta identificación y clasificación de las moras en sus diferentes estados de madurez. Las imágenes de prueba confirmaron la capacidad del modelo para diferenciar los colores utilizando los rangos HSV establecidos.

Métricas de rendimiento

Estas gráficas muestran el progreso del modelo YOLOv8m durante el proceso de entrenamiento y validación. Se presentan diversas métricas clave, como las pérdidas y medidas de precisión, que permiten evaluar la mejora del modelo a lo largo del tiempo.

En la Figura 71 observamos la gráfica de pérdidas de entrenamiento para las cajas delimitadoras (train/box_loss) que muestra la evolución de la pérdida asociada a la predicción de las cajas delimitadoras (bounding boxes). Existe una disminución constante desde aproximadamente 3.0 hasta cerca de 0.7, indicando que el modelo mejora en la precisión al predecir las posiciones y tamaños de las cajas que encierran los objetos. La pérdida asociada a la clasificación de los objetos dentro de las cajas delimitadoras (train/cls_loss) se reduce de un valor inicial alto (~20) a menos de 1.0, mostrando que el modelo se vuelve más preciso en identificar las clases de los objetos. La pérdida de la distribución focal (train/dfl_loss) mide la precisión con la que el modelo asigna pesos a diferentes áreas dentro de las cajas delimitadoras. Esta pérdida decrece de 1.6 a alrededor de 0.8, contribuyendo a mejorar la calidad general de las predicciones. Las Métricas de entrenamiento refleja la precisión del modelo (metrics/precision(B)) esta métrica mejora de 0.6 a casi 0.9, indicando que el modelo identifica correctamente más objetos sin confundirlos con elementos irrelevantes. En la muestra de la tasa de recuperación (metrics/recall(B)) se incrementa de 0.5 a aproximadamente 0.75, indicando que el modelo reduce los falsos negativos. En Pérdidas de validación (val/box_loss, val/cls_loss, val/dfl_loss) muestran tendencias similares a las de entrenamiento, con reducciones significativas desde valores iniciales más altos hacia niveles más bajos. Esto sugiere que el modelo se generaliza bien a los datos de validación. Las Métricas de validación se observa que las métricas de

precisión media promedio al umbral IoU del 50% (metrics/mAP50(B) aumenta progresivamente hasta alcanzar un valor cercano a 0.8, mostrando que el modelo mejora su rendimiento para detectar objetos correctamente y la precisión media promedio en un rango de umbrales (50% a 95%) (metrics/mAP50-95(B)) sube consistentemente, alcanzando alrededor de 0.6, indicando una mejora en la capacidad del modelo para detectar objetos en diferentes escenarios.

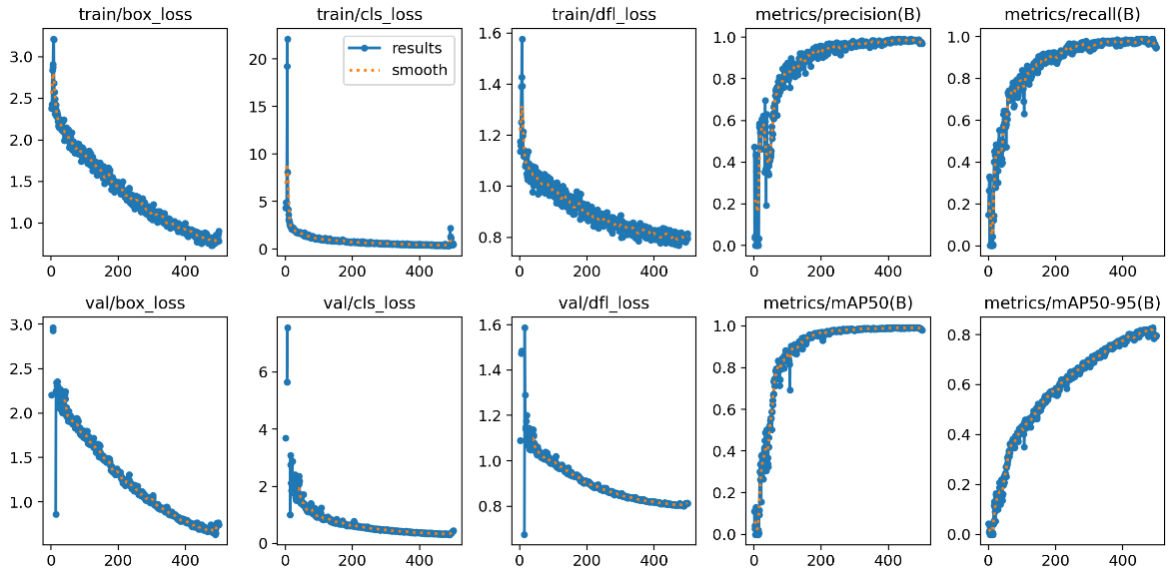

Figura 71. Métricas de rendimiento del modelo entrenado (Detección de moras).

En general, estas gráficas demuestran que el modelo YOLOv8m mejora significativamente en su capacidad de detección de objetos, tanto en términos de precisión como de generalización, a medida que avanza el entrenamiento y la validación.

Matriz de Confusión

La matriz de confusión normalizada presentada refleja el desempeño del modelo YOLOv8m al clasificar las diferentes clases de interés en el conjunto de datos evaluado. Esta herramienta es esencial para comprender cómo el modelo distribuye sus predicciones entre las clases verdaderas y cómo se comporta frente al ruido o errores en los datos.

En la Figura 72 se observa que las clases principales (1, 2, 3) la diagonal principal muestra los valores de predicción correcta para cada clase. En la Clase 1 se logró una precisión del 100% (1.00), indicando que todos los objetos de esta clase fueron correctamente identificados. La Clase 2 alcanzó un valor de 0.99, con un leve desbalance hacia otras clases. Para la Clase 3 se presenta un desempeño similar con un valor de 0.99 en la diagonal principal. Esto muestra que el modelo tiene un alto nivel de confianza y precisión en estas clases principales. Los errores de clasificación los tenemos en la Clase 2 que se registra una confusión con un valor de 0.70 hacia la clase 3, lo que indica que el modelo podría tener dificultades en la diferenciación entre estas clases debido a similitudes visuales o características compartidas.

En la Clase background a pesar de representar áreas sin objetos, algunas predicciones incorrectas la clasifican como 1 o 3, con valores de 0.15 en ambos casos. Esto puede indicar que ciertos patrones en los datos de fondo son similares a los objetos.

En general podemos decir que as clases principales (1, 2, 3) tienen un desempeño robusto con valores en la diagonal principal cercanos a 1.0. Sin embargo, los errores observados en clases cercanas o en el fondo (background) resaltan la necesidad de ajustar el modelo o mejorar los datos de entrenamiento para reducir ambigüedades.

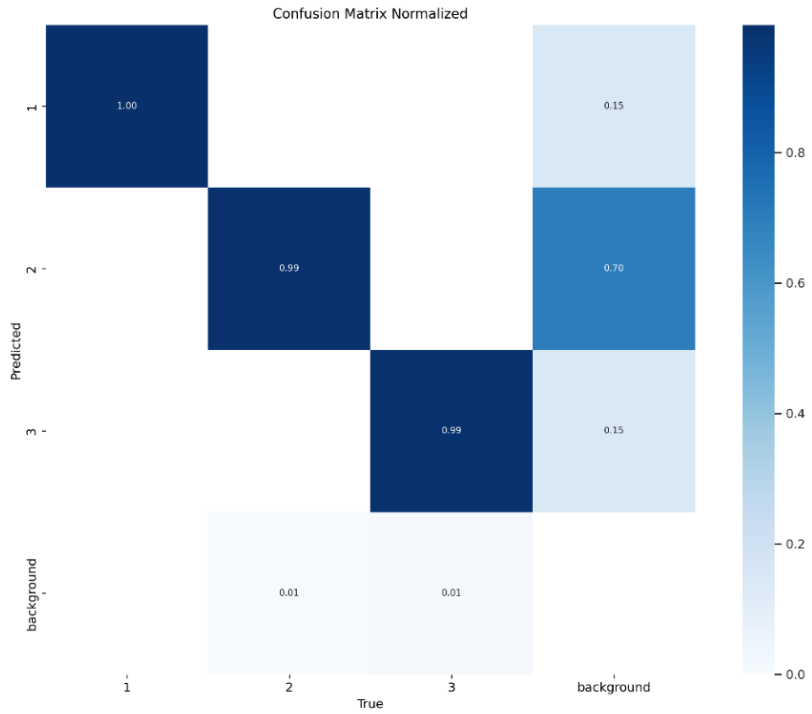

Figura 72. Matriz de Confusión.

Matriz de correlación

La matriz de correlación ofrece una representación gráfica que facilita la visualización de las interrelaciones lineales entre varias métricas y parámetros empleados en el proceso de entrenamiento y validación del modelo YOLOv8m. En la celda se tiene un rango de -1 a 1 que representa el coeficiente de correlación entre métricas. Cuando un valor se aproxima al valor unitario existe una correlación positiva, en cambio sí se aleja del valor unitario se considera negativa.

En la Figura 73 se analiza los valores de correlación entre pérdidas de entrenamiento y métricas de validación y se concluye que las pérdidas de entrenamiento (train/box_loss, train/cls_loss, train/dfl_loss) muestran una fuerte correlación negativa con las métricas de precisión (metrics/precision(B)), recall (metrics/recall(B)), mAP50 (metrics/mAP50(B)) y mAP50-95 (metrics/mAP50-95(B)). Esto indica que a medida que las pérdidas disminuyen, el desempeño del modelo mejora considerablemente.

La Relación entre métricas de validación y entrenamiento, las pérdidas de validación (val/box_loss, val/cls_loss, val/dfl_loss) también están altamente correlacionadas con las métricas de desempeño. Esto refuerza que un buen ajuste en el conjunto de validación refleja mejoras generales del modelo.

Los parámetros relacionados con la tasa de aprendizaje (lr/pg0, lr/pg1, lr/pg2) tienen correlaciones significativas con las métricas de desempeño y las pérdidas, indicando su papel fundamental en el ajuste del modelo.

Las métricas como precisión, recall, mAP50 y mAP50-95 están altamente correlacionadas entre sí (valores cercanos a 1), lo cual es esperado, dado que todas evalúan diferentes aspectos del mismo rendimiento del modelo.

En la evolución temporal podemos decir que la métrica de tiempo muestra una correlación moderada con las pérdidas y métricas, lo que sugiere que los cambios en el modelo a lo largo de las épocas influyen directamente en su desempeño.

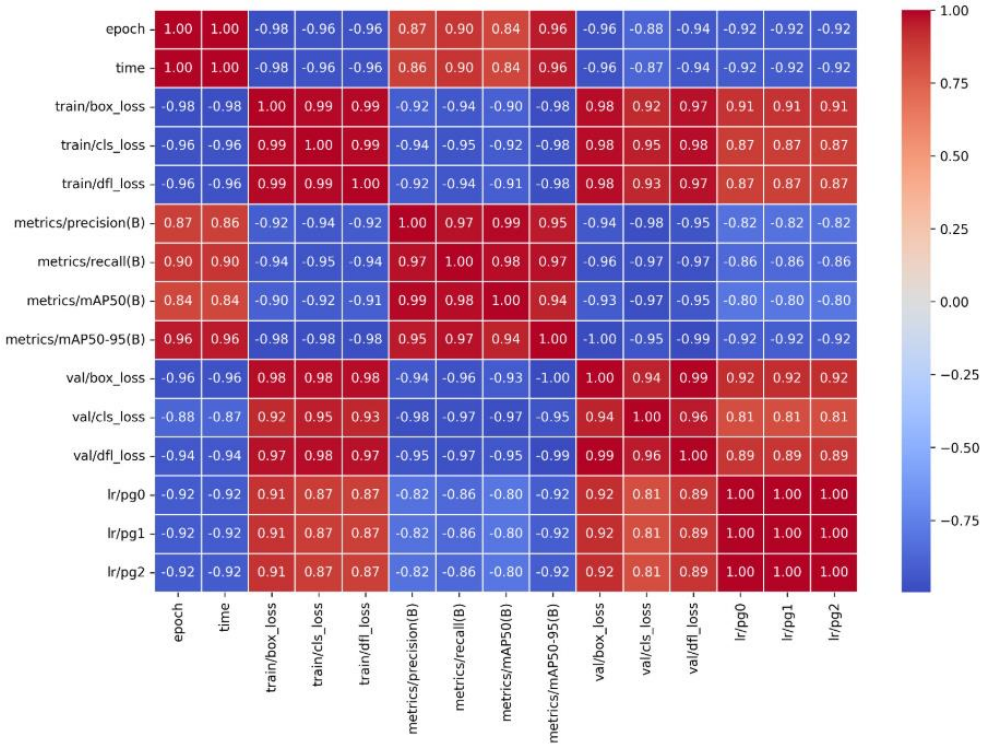

Figura 73. Matriz de Correlación.

Predicción por lote de validación

La Figura 74 representa la predicción del modelo sobre un lote de validación utilizando imágenes de cultivos de mora. Este resultado ilustra cómo el modelo detecta y clasifica las moras en las diferentes etapas de madurez (por ejemplo, verde, roja, y madura) mediante cuadros delimitadores (bounding boxes) y etiquetas numéricas. Cada cuadro delimitador está asociado a un número que representa la clase detectada:

- ✓ **Clase 0:** Mora verde.
- ✓ **Clase 1:** Mora roja.
- ✓ **Clase 2:** Mora madura.

Estas etiquetas permiten evaluar la capacidad del modelo para distinguir entre las diferentes clases según la etapa de maduración. Estas predicciones son fundamentales para evaluar el rendimiento del modelo con datos no vistos previamente (datos de validación). Los resultados pueden usarse para calcular métricas como la precisión media (mAP) o para analizar casos específicos de falsos positivos y negativos.

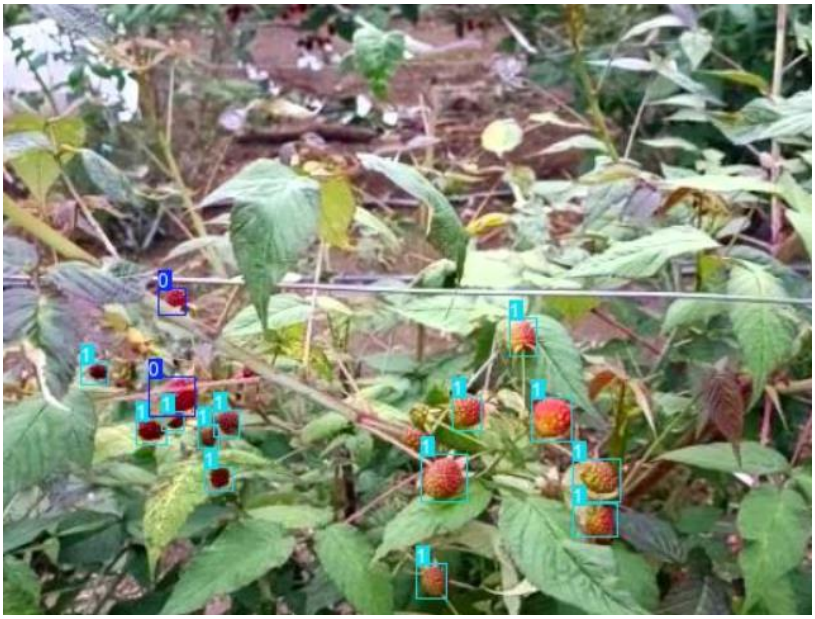

Figura 74.Predicción por lote modelo detección de moras

Representación de la curva de P-R

La gráfica de P-R muestra cómo se relacionan la precisión (que evalúa la exactitud de las predicciones) y el recall (que mide la capacidad del modelo para identificar todas las instancias pertinentes).

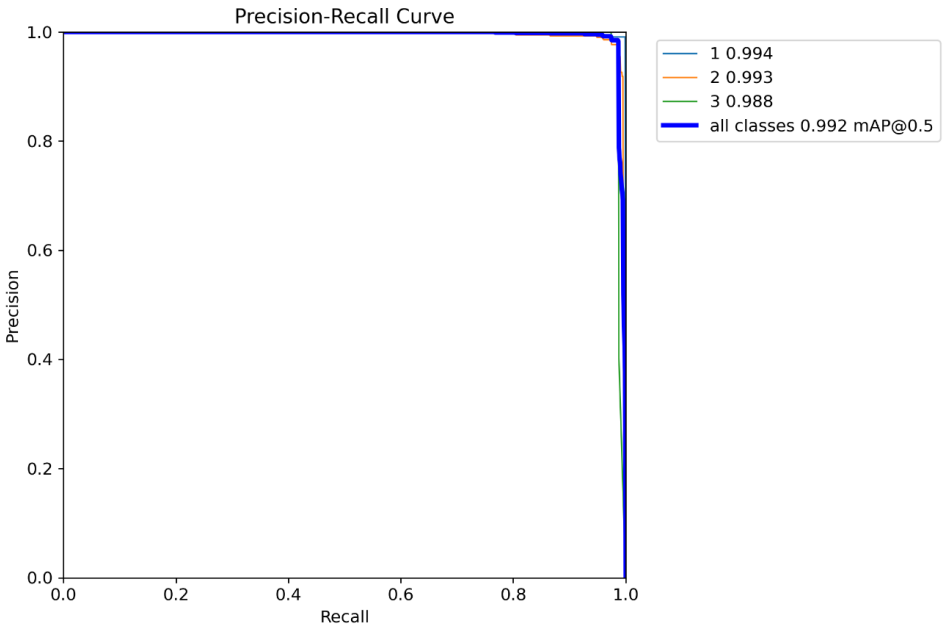

Figura 75. Curva de Precisión y Recuperación

En la Figura 75 las líneas correspondientes a las clases (azul para clase 1, naranja para clase 2 y verde para clase 3) son casi idénticas y están concentradas cerca del punto ideal de precisión y recall igual a 1. Esto indica que el modelo tiene un desempeño muy consistente y robusto entre las clases. La combinación de valores altos de precisión y recall, junto con un mAP@0.5 promedio de 0.992, refleja un modelo bien entrenado y optimizado para las tareas de clasificación y detección.

Este gráfico refuerza la idea de que el modelo es adecuado para la tarea de estimación de producción de mora, ya que ofrece predicciones confiables y consistentes para las diferentes clases de objetos.

Curva de Precisión

La curva de precisión en función de la confianza (Precision-Confidence Curve) presentada es un análisis gráfico de cómo varía la precisión del modelo con respecto al umbral de confianza en sus predicciones. Este tipo de gráfico es particularmente útil para ajustar los valores de confianza que maximizan el rendimiento del modelo.

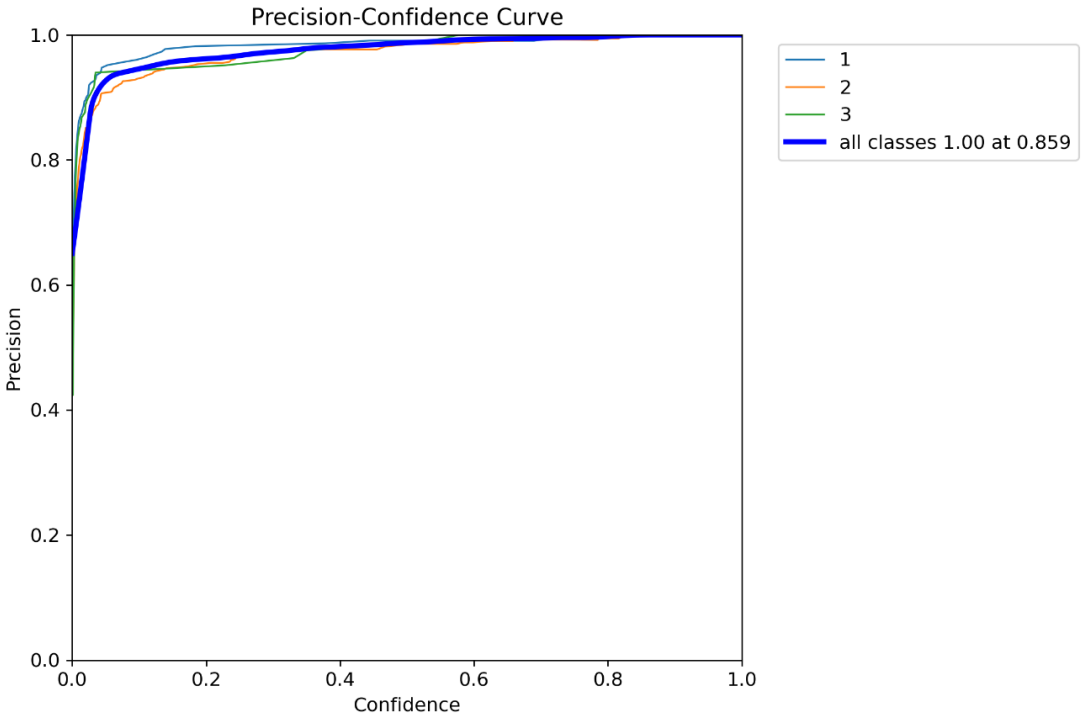

Figura 76. Curva de Precisión

En la Figura 76 las líneas de colores representan las diferentes clases del modelo (1, 2 y 3). Todas presentan un comportamiento similar, alcanzando valores altos de precisión a medida que el umbral de confianza incrementa. La línea azul más gruesa representa el rendimiento promedio de todas las clases.

En este caso, el modelo logra una precisión combinada de 1.00 cuando la confianza alcanza aproximadamente 0.859. Una línea más plana y alta es ideal, ya que indica que la precisión es alta incluso con umbrales más bajos.

- ***Resultados modelo VGG16***

El modelo VGG16 tiene como resultado una exactitud del 86% en la validación y en entrenamiento es de 84%, se obtiene valores altos de exactitud como se muestra en la Figura 77 el crecimiento exponencial del modelo.

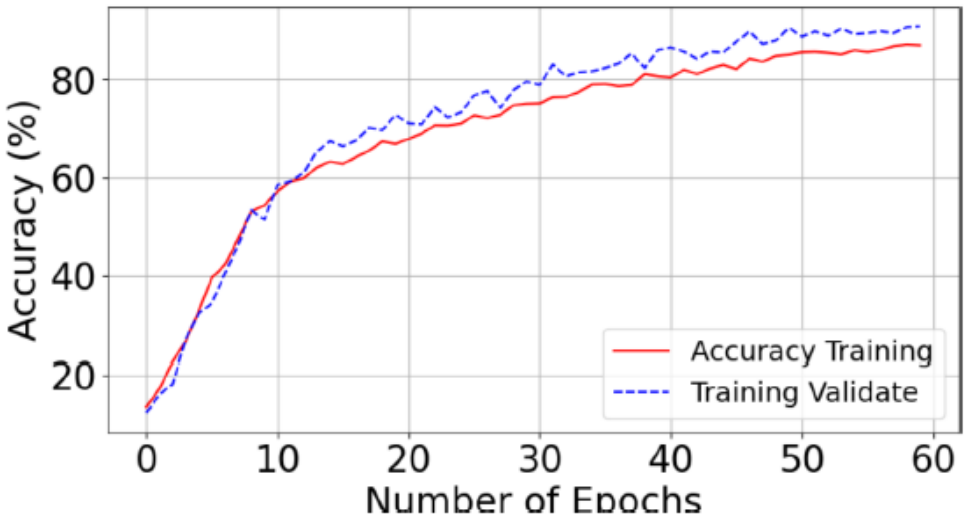

Figura 77. Curva de exactitud modelo VGG16.

En la Figura 78 de muestra las métricas de perdidas (loss) lo que corresponde a perdidas bajas, la pérdida disminuye constantemente y converge hacia valores muy bajos de perdidas, el modelo entrenado mejora conforme avanzas las iteraciones, para la validación se tiene un valor de 0.4135 en cambio el valor de pérdida del entrenamiento es 0.337.

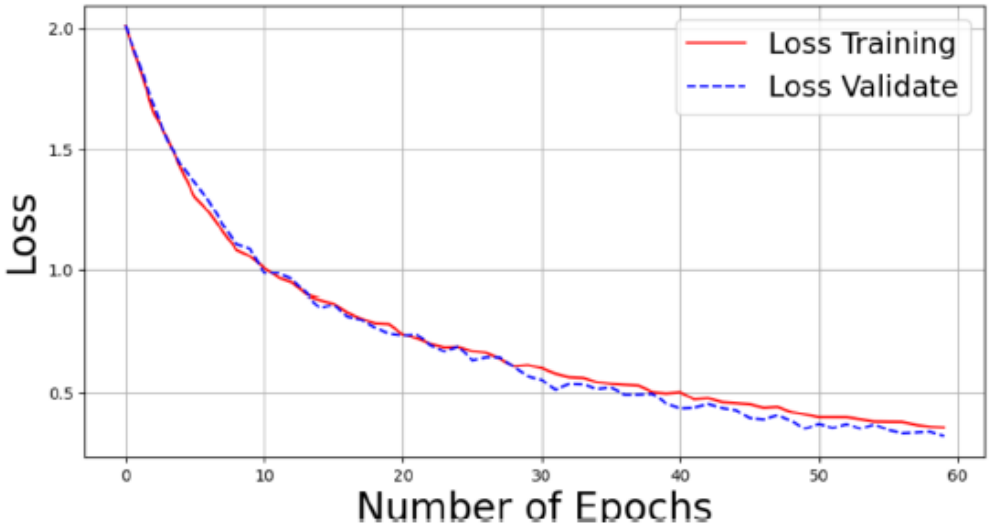

Figura 78. Curva de pérdidas del modelo VGG16.

g. Comparación de modelos

En la siguiente Tabla 25 se comparan los resultados del modelo de Yolo y el modelo VGG16, por lo que se determina que el modelo Yolov8m tiene una exactitud del 95%, con una pérdida de 0.18, por lo que comparado con el modelo al modelo VGG16 es mucho más optimo y es por ello por lo que el modelo de Yolo es seleccionado.

Tabla 25. Comparación de modelos.

Modelo	Exactitud	Perdida
Yolov8m	95%	0.18
VGG16	86%	0.4135

h. Sistema de gestión de base de datos

La base de datos utilizada es MongoDB que almacena los datos del modelo ya entrenado anteriormente, donde empieza guardando los datos de la detección de moras y las clasifica por medio de ID según el estado de maduración de la mora (verde, roja y madura o morada) como se muestra en la Figura 79.

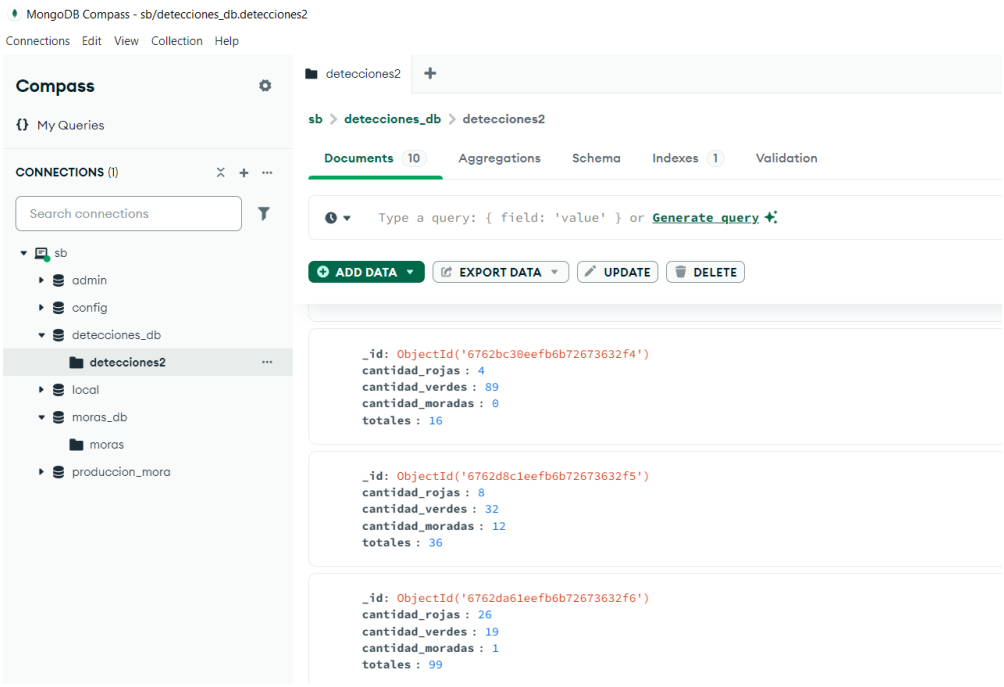

Figura 79. Base de datos de las moras en MongoDB.

En la Figura 80 se muestra un diagrama del sistema de base de datos en la cual la tabla se llama detecciones, este registro funciona como un listado que establece los estados de maduración de las moras. Incluye las siguientes columnas: **ID_Estado (Pk):** Es un identificador asignado a cada estado: verde, rojo y morado.

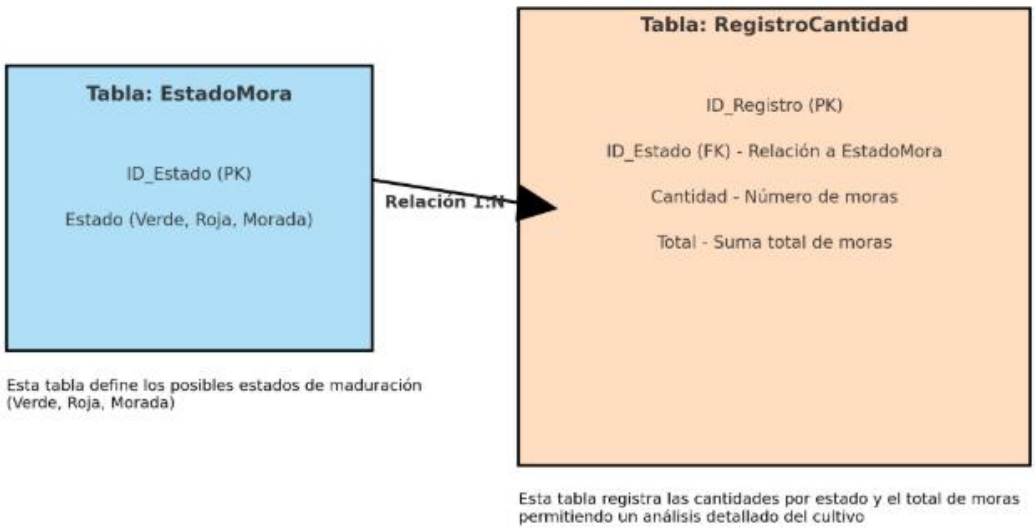

Figura 80. Diagrama entidad-relación del sistema de base de datos.

La tabla de registro de cantidad de moras documenta las cantidades de moras organizada según su estado de maduración, incluyendo el número total de las moras. Las columnas incluyen: **ID_Registro (PK):** que actúa como un identificador para cada entrada. **ID_Estado (FK):** hace referencia al ID_Estado de la tabla estado de mora, por lo que se crea una conexión entre las tablas.

La relación entre tablas: es una relación 1:N (uno a muchos), cada estado de mora en la tabla se vincula a varios registros de la cantidad de moras.

i. Interfaz gráfica y funcionalidades adicionales

Cuando los datos ya están almacenados en la base de datos, se ha desarrollado la interfaz con Flask que es diseñado para ser una interfaz web amigable e intuitiva con el usuario, además es de gran ayuda para el desarrollador ya que tiene la libertad de elegir bibliotecas y extensiones adicionales según sus necesidades.

Por lo que se desarrollado una interfaz de usuario para la visualización en el microordenador NVIDIA Jetson Nano.

Sección que define las rutas para la interfaz HTML y funcionalidades como sumar filas o mostrar tablas.

```
@app.route('/')
def index():
    return send_file(os.path.join(os.path.dirname(__file__), 'index.html'))

@app.route('/tabla')
def tabla():
    documentos = collection.find()
    return render_template("tabla.html", documentos=documentos)

@app.route('/sumar_filas', methods=['POST'])
def sumar_filas():
    datos = request.json.get('datos', [])
```

✓ Sirve los archivos HTML y CSS necesarios para la interfaz.

✓ Implementa una funcionalidad para sumar filas de datos enviadas desde la interfaz.

✓ Guarda los datos dentro de la base de datos.

j. Ejecución de la aplicación

```
if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

Inicia el servidor Flask en el puerto 5000, habilitando el modo de depuración para pruebas. En el sistema de la interfaz gráfica tiene un sistema de seguridad para navegar en la interfaz web conectada a la base de datos de donde se recibirá la información de datos de las moras, así cumpliendo con el objetivo 4, como se observa en la Figura 81.

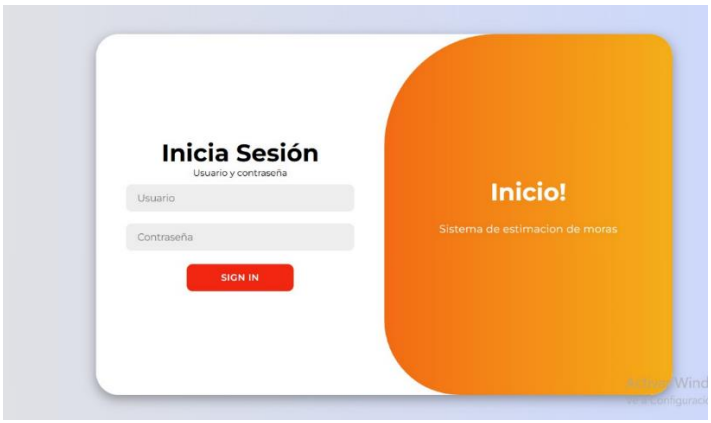

Figura 81. Interfaz de autenticación de usuario.

Después de ingresar el usuario y la contraseña se despliega una ventana donde se tiene 3 botones, el primero es para reproducir el video que se obtiene con el dron y va reconociendo las moras en tiempo real, el segundo es para detener el video una vez acabo la inspección con el dron en el cultivo de mora, y finalmente el botón de visualización de las tablas con el conteo de moras como se muestra en la Figura 82 .

Figura 82. Botones principales de la interfaz gráfica.

Al pulsar el botón reproducir video observamos la región de la mora (ROI), la etiqueta asignada (track_id) y el color detectado por el modelo en tiempo real como ve observa en la Figura 83.

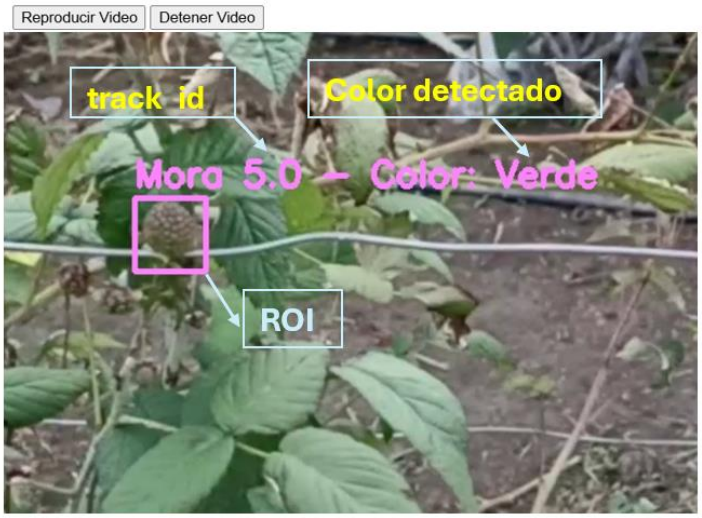

Figura 83. Detección y conteo de la mora.

En la Figura 84 se observa la tabla de conteo de las moras con la detección del video reproducido anteriormente, en la tabla se observa la cantidad de moras verdes, rojas y maduras o moradas totales, además de 4 botones adicionales.

Tabla de Detecciones

Cantidad Rojas	Cantidad Verdes	Cantidad Moradas	Totales
89	50	38	177
4	89	0	16
8	32	12	36
26	19	1	99
5	30	5	18
51	50	13	64
8	30	2	10
30	20	1	31
39	25	1	40
5	13	0	18

Calcular

Mostrar/Ocultar Gráficos

Volver al inicio

Prediccion Montecarlo

Figura 84. Tablas de detección de moras.

El primero es para calcular el costo y el peso de las moras, el segundo botón es de mostrar las gráficas del conteo de moras, el tercero es un botón de volver al inicio y el ultimo botón es para predecir las moras en la próxima cosecha con el método de Montecarlo.

En el primer botón del cálculo, se calcula el peso total de las moras moradas y el costo de las moras moradas como se muestra en la Figura 85. En esta sección, la simulación incluye una extensión para estimar costos asociados a las muestras detectadas.

Con base en las moras detectadas y los datos de producción, se permite calcular pérdidas económicas debido a las plagas y heladas, considerando factores como el precio promedio por mora y los costos indirectos de mitigación.

Resultados por categoría:

Cantidad Rojas: 265 unidades

Estimación de Peso: 1749.00 g

Cantidad Verdes: 358 unidades

Estimación de Peso: 2362.80 g

Cantidad Moradas: 73 unidades

Estimación de Peso: 481.80 g

Estimación de Costo: 4.23$

Figura 85. Cálculo de costo y peso de moras.

En la Figura 86 se muestra las gráficas del peso de las moras y el costo de estas.

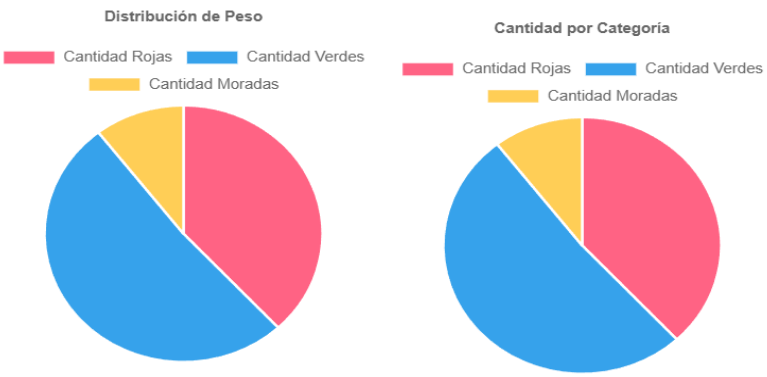

Figura 86. Graficas de peso y costo de las moras.

El botón de predicción de mora como se muestra en la Figura 87 con el Método de Montecarlo se usa poniendo un porcentaje de probabilidad de plaga y helada ya que son dos factores más importantes en pérdidas de moras. El algoritmo implementa fórmulas específicas para calcular el daño causado por plagas y heladas, dependiendo de las probabilidades ingresadas y las características de las frutas (colores). El resultado es presentado al usuario en la interfaz, proporcionando una herramienta de análisis predictivo robusta.

```
def montecarlo_simulation(prob_plaga, prob_helada, cantidad_rojas,
    cantidad_verdes, cantidad_moradas, totales, num_simulaciones=10000):
    total_frutas_afectadas = 0
    for _ in range(num_simulaciones):
        afectadas_por_plaga = (prob_plaga * 0.30 * cantidad_rojas) +
        (prob_plaga * 0.10 * cantidad_moradas)
        afectadas_por_helada = prob_helada * 0.20 * cantidad_verdes
        frutas_afectadas = afectadas_por_plaga + afectadas_por_helada
        total_frutas_afectadas += frutas_afectadas
    promedio_afectadas = total_frutas_afectadas / num_simulaciones
    porcentaje_afectadas = (promedio_afectadas / totales) * 100
    return round(porcentaje_afectadas, 2)
```

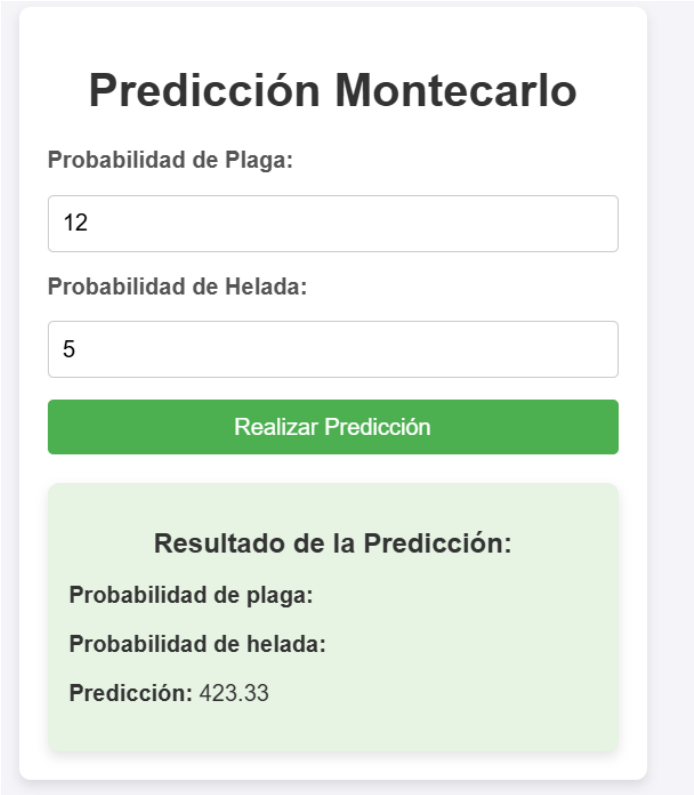

Figura 87. Interfaz de predicción de la cantidad de la futura cosecha.

En la Figura 87 se muestra una simulación del método de Montecarlo en la cual se basa en un valor que representa la probabilidad estimada (en porcentaje) de que se produzca una plaga en el cultivo, en este caso se ingresa de manera manual un porcentaje del 12% ya que hay una probabilidad moderada de que ocurra plagas que afecten al cultivo de mora, el 5% indica que existe una baja posibilidad de que una helada afecte el cultivo.

El resultado final del método de Montecarlo representa la predicción numérica de la media de 423.33 representa una media de producción de mora en este caso son el total del lote de moras moradas en la plantación. en la fila

3.3.5 Implementación del sistema

La implementación del sistema se emplea el dron DJI Mini 2 donde se captura las imágenes y video en tiempo real para detectar las moras, el control del dron para el control y su respectiva monitorización por medio de un celular, la Jetson Nano donde se procesa los datos, el monitor donde está la base de datos y la interfaz gráfica para visualizar los resultados de la clasificación de las moras según su estado de maduración, por último el teclado y el mouse completan el sistema como se muestra en la Figura 88.

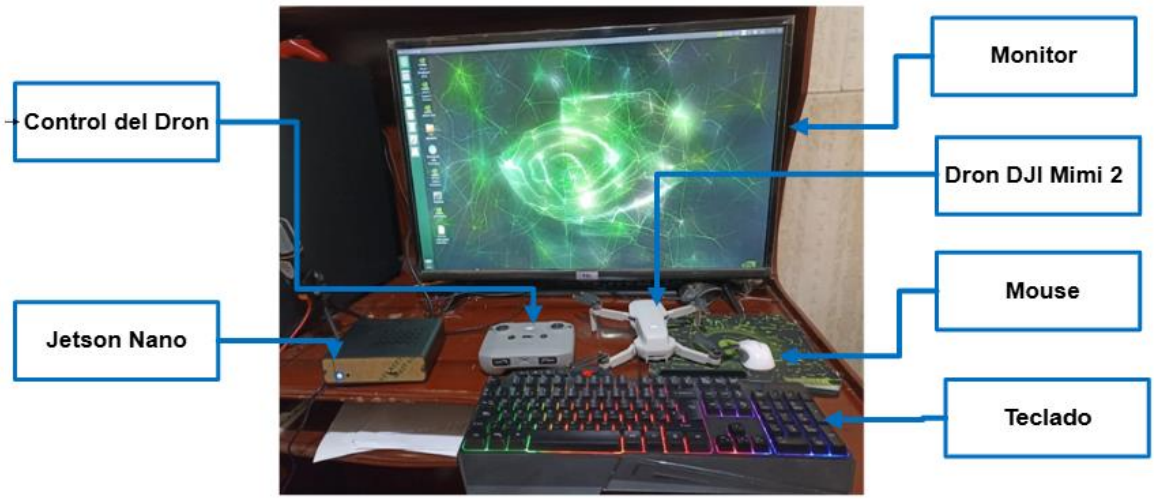

Figura 88. Esquema prototipo del sistema.

a. Instalación del sistema operativo

Para acoplar la Jetson Nano al sistema operativo, el primer paso es descargar la versión más actual de NVIDIA JetPack, por medio de Etcher se graba la imagen descargada en una tarjeta microSD para posteriormente colocarla en la Jetson Nano, después se conecta al monitor para iniciar el sistema para la interfaz de la Jetson Nano como se muestra en la Figura 89 .

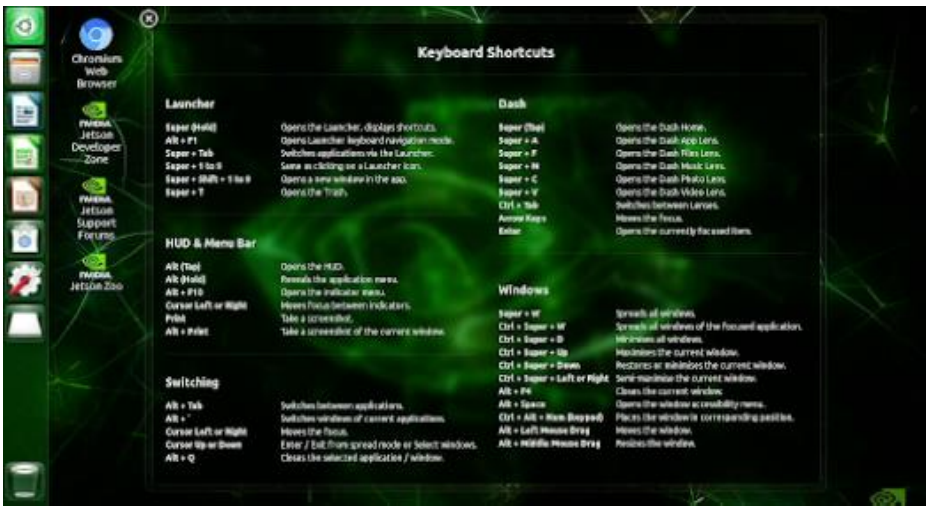

Figura 89. Interfaz Jetson Nano.

b. Configuración del Entorno de Desarrollo

Para desarrollar el sistema en la Jetson Nano se activa CUDA que permite utilizar la GPU para acelerar el sistema de los modelos entrenados que son muy pesados. CUDA no viene activado por defecto, se debe clonar un repositorio para modificar algunas configuraciones.

En el repositorio se encuentra un script en la cual se compila y se instala OpenCV para el procesamiento de imágenes con el siguiente comando: **nano_build_opencv.** Es de gran importancia ajustar las versiones de CUDA y cuDNN en el script para que sean compatibles con la arquitectura de la Jetson Nano.

El proceso de compilación tarda alrededor de una 6 a 7 horas como se muestra en la Figura 90.

Figura 90. Configuración de OpenCV en la Jetson Nano.

c. Implementación de bibliotecas requeridas

Para tener un óptimo uso de la Jetson Nano y tener un buen sistema se necesita instalar algunas librerías importantes tales como:

- ✓ FilterPy **s**e instala con el siguiente comando: s**udo pip3 install filterpy.**
- ✓ Ultralytics se instala con el siguiente comando: **pip3 install ultralytics.**
- ✓ Flask se instala con el siguiente comando: **pip3 install flask.**

- ***Pruebas en tiempo real del sistema de estimación de mora***

Prueba 1

El desarrollo de la primera prueba de funcionalidad del sistema de estimación de producción de mora se lleva a cabo en cuatro etapas principales, descritas de manera cronológica a continuación:

Acceso a la interfaz del sistema

El usuario realiza el acceso a la interfaz gráfica del sistema mediante un navegador web compatible, ingresando las credenciales previamente configuradas. La interfaz presenta las opciones principales del sistema, como la configuración del dron, la selección del área de cultivo y el monitoreo en tiempo real.

Configuración y planificación de trayectorias

A través de la interfaz, se seleccionan las áreas específicas del cultivo a analizar. El sistema establece las coordenadas GPS y genera una planificación de trayectoria optimizada mediante la herramienta Rainbow for DJI, asegurando una cobertura completa del área de interés.

Adquisición de datos e inicio del análisis

En la Figura 91 se muestra la adquisición de datos en este caso son las moras.

Figura 91. Inicio de la cámara en la primera prueba de funcionamiento del sistema.

El dron DJI Mini 2 realiza el despegue automático y sigue la ruta establecida. Durante el vuelo, se captura un conjunto de fotogramas del cultivo en tiempo real. En la Figura 92 se observa como estas imágenes son enviadas al sistema para su procesamiento, donde se utilizan algoritmos basados en YOLOv8m y técnicas de visión artificial, como el análisis de color HSV, para detectar y clasificar moras en diferentes estados de madurez.

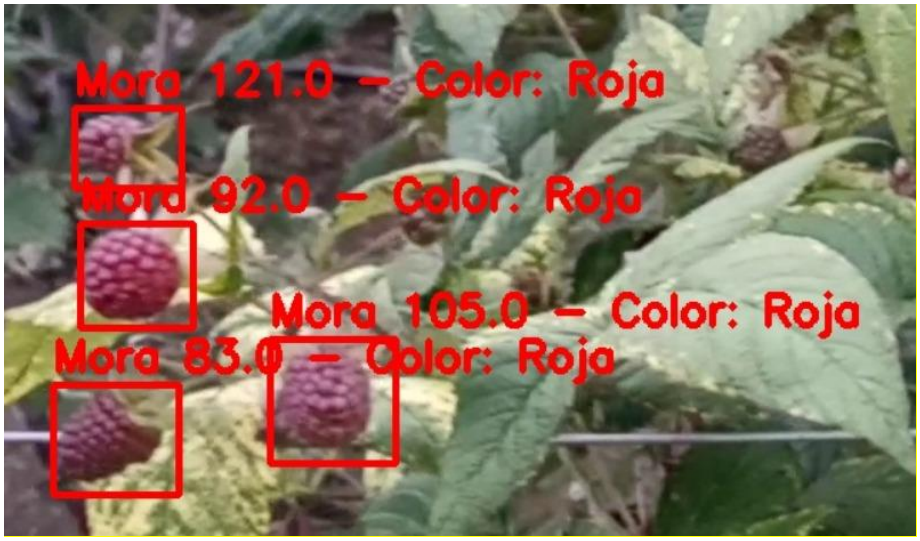

Figura 92. Detección en tiempo real.

Resultados y visualización

En la Tabla 26 los resultados del conteo total de moras por estado (verdes, rojas y moradas), que se muestran en la interfaz gráfica.

Tabla 26. Resultados de la primera prueba.

Moras Verdes	Moras Rojas	Moras Moradas	Total, de Moras
89	50	38	177

En el interfaz, una vez almacenado los datos en MongoDB, se realiza la estimación de la producción como se observa en la Figura 93 .

Resultados por categoría:

Cantidad Rojas: 89 unidades

Estimación de Peso: 58.74 g

Cantidad Verdes: 50 unidades

Estimación de Peso: 33.00 g

Cantidad Moradas: 38 unidades

Estimación de Peso: 33.44 g

Estimación de Costo: $1.67

Figura 93. Cálculo y resultado de la estimación de la producción.

Se observa en la Figura 94 con la ayuda de la predicción del clima podemos calcular la probabilidad de heladas. En meteorología, una helada ocurre cuando la temperatura desciende entre 3 °C y 0 °C. Para el mes se tiene una probabilidad de al menos 1 día de helada.

Figura 94. Predicción meteorológica del del clima

Con los datos históricos, calcula la probabilidad de heladas usando esta fórmula:

$$P(helada) = \frac{\#\ de\ días\ en\ enero\ con\ temperatura\ entre\ 3\ y\ 0\ °C}{\#\ total\ de\ dias\ en\ enero} \quad (1)$$

$$P(helada) = \frac{1}{31} * 100$$

$$P(helada) = 0{,}033 * 100$$

$$P(helada) = 3{,}3\ \%$$

En la Figura 95 la simulación de Montecarlo integrada al sistema permite la evaluación de posibles pérdidas de producción considerando factores como plagas o condiciones climáticas adversas. En esta prueba observamos que la predicción para esta fila con ayuda de los datos históricos es de una media de 113.89 moras moradas para la cosecha del mes próximo.

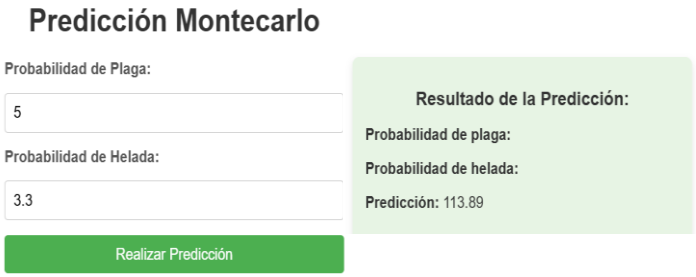

Figura 95. Simulación de Montecarlo en la interfaz del sistema.

3.3.6 Resultados de las pruebas de funcionamiento

En la Tabla 27 se presentan las pruebas de funcionamiento del conteo de moras moradas. Se detalla la comparación entre los resultados obtenidos mediante la inspección con el sistema de estimación y los resultados derivados de la inspección manual, considerando un total de 18 pruebas realizadas en la fila.

Tabla 27. Pruebas de funcionamiento.

#Prueba	Hora	Método manual	Kg/Fila	Modelo (Lado izquierdo)	Modelo (Lado derecho)	Modelo (Total)	Kg/Fila
1	8:30	185	1,63	89	67	156	1,37
2	8:30	253	2,23	156	50	206	1,81
3	8:30	70	0,62	29	32	61	0,54
4	8:30	104	0,92	45	39	84	0,74
5	8:30	209	1,84	142	30	172	1,51
6	8:30	90	0,79	70	9	79	0,70
7	8:30	224	1,97	133	28	161	1,42
8	8:30	160	1,41	56	80	136	1,20
9	8:30	270	2,38	134	89	223	1,96
10	8:30	145	1,28	103	27	130	1,14
11	8:30	86	0,76	42	35	77	0,68
12	8:30	165	1,45	87	55	142	1,25
13	8:30	223	1,96	117	77	194	1,71
14	8:30	111	0,98	56	45	101	0,89
15	8:30	242	2,13	135	88	223	1,96
16	8:30	165	1,45	79	73	152	1,34
17	8:30	238	2,09	96	117	213	1,87
18	8:30	158	1,39	84	67	151	1,33
	Media	**172**	**1,5**	**92**	**56**	**148**	**1,3**

En la siguiente Tabla 28 se muestra la estimación de producción por fila, se establece como primeros datos las 3 primeras parcelas o surcos de las cuales se emplea el modelo automático diseñado en la cual realiza el conteo de las moras moradas y se estima un peso 8.8 gramos como media, de las cuales se multiplican el total de moras por el peso promedio antes dicho.

Tabla 28. Tabla de estimación de producción por fila (kg/fila)

Fila	Moras Moradas (Modelo Automático)	Peso Total (kg)
1	327	2,87
2	141	1,51
3	184	2
Media	**217**	**2,12**

Estimación para las 21 filas

En la Tabla 29 realizamos el cálculo de la estimación para toda la parcela

$$\boldsymbol{Estimación\ total\ (kg) = Promedio\ por\ fila\ (kg) \times \#\ de\ surcos} \qquad (\mathbf{2})$$

$$\boldsymbol{Estimación\ total\ (kg)} = 2{,}12kg \times 21$$

$$Estimación\ total\ (kg) = 44{,}72\ kg$$

Tabla 29. Tabla de estimación para las 21 filas

Estimación para las 21 filas	
Producción por fila (Kg):	2,12
Producción por 21 filas (Kg):	44,72
Numero de Baldes:	6
Valor total en USD:	$ 30

Los resultados de estimación para las 21 Filas de la parcela de cultivo de mora es que la producción por fila es de aproximadamente 2,12 kilogramos por fila. La producción total de las 21 filas es de 44,72 kilogramos que equivale a 6 baldes.

Predicción de cosecha

En la Tabla 30 se muestra la predicción de la cosecha del cultivo de mora establecida por medio del modelo que cuenta por el lado derecho y el lado izquierdo de las parcelas

de las moras y compara con un conteo manual de las moras moradas o maduras en este caso se lo realizo al momento de la cosecha de la mora que es los martes a las 8:30.

Tabla 30. Pruebas de funcionamiento para la predicción de cosecha.

#Prueba	Hora	Método manual		Modelo (Lado izquierdo)		Modelo (Lado derecho)		Modelo Automático (Total)	
		Moras Verdes	Moras Rojas	Moras Verdes	Moras Rojas	Moras Verdes	Moras Rojas	Moras Verdes	Moras Rojas
1	8:30	131	118	67	58	45	46	112	104
2	8:30	154	135	78	62	56	50	134	112
3	8:30	177	151	68	68	49	65	117	133
4	8:30	203	167	75	72	60	75	135	147
5	8:30	224	187	44	98	100	75	144	173
6	8:30	252	204	75	101	98	78	180	179
7	8:30	278	222	80	109	110	80	190	189
8	8:30	304	239	90	110	117	85	210	195
9	8:30	328	256	88	118	128	98	220	216
10	8:30	354	275	98	87	135	105	240	192
11	8:30	379	292	103	113	150	99	253	212
12	8:30	375	230	137	110	103	100	259	210
13	8:30	390	250	145	140	103	78	260	218
14	8:30	405	309	185	136	138	120	323	256
15	8:30	480	427	234	220	132	160	380	380
16	8:30	472	340	177	130	198	140	375	270
17	8:30	358	279	186	105	103	98	289	203
18	8:30	280	217	105	80	97	90	202	170
	Media	**308**	**239**	**113**	**107**	**107**	**91**	**224**	**198**

a. Cálculo de error

Para calcular el valor absoluto del conteo de moras se separan por el estado de maduración de la mora en este caso mora verde, mora roja y mora madura o morada mediante las moras detectadas por el sistema implementado y por el conteo manual y validado por un ingeniero agrónomo.

Por lo que mediante la siguiente formula se puede determinar el error absoluto:

$$\boldsymbol{Error\ absoluto = |Valor\ real - Valor\ obtenido|} \qquad (\mathbf{3})$$

Por otro lado, para el cálculo del error relativo se aplica la siguiente fórmula:

$$\boldsymbol{Error\ relativo} = \left|\frac{\boldsymbol{Valor\ teórico - Valor\ medido}}{\boldsymbol{Valor\ teórico}}\right| * \mathbf{100} \qquad (\mathbf{4})$$

Para los cálculos de error absoluto y error relativo se toma en cuenta los promedios del conteo manual y el conteo con el sistema de las moras verdes, rojas y maduras de la Tabla 27 y la Tabla 30.

- ***Cálculo de las moras verdes***

Error absoluto

$$\boldsymbol{Error\ absoluto} = |\boldsymbol{Valor\ real} - \boldsymbol{Valor\ obtenido}|$$

$$Error\ absoluto = |308 - 224|$$

$$Error\ absoluto = 84$$

Error relativo

$$\boldsymbol{Error\ relativo} = \left|\frac{\boldsymbol{Valor\ teórico - Valor\ medido}}{\boldsymbol{Valor\ teórico}}\right| * \mathbf{100}$$

$$Error\ relativo = \left|\frac{308-224}{308}\right| * 100$$

$$Error\ relativo = 27.27\%$$

- ***Cálculo de las moras rojas***

Error absoluto

$$\boldsymbol{Error\ absoluto} = |\boldsymbol{Valor\ real} - \boldsymbol{Valor\ obtenido}|$$

$$Error\ absoluto = |239 - 198|$$

$$Error\ absoluto = 41$$

Error relativo

$$Error\ relativo = \left|\frac{239 - 198}{239}\right| * 100$$

$$Error\ relativo = 17{,}15\%$$

- ***Cálculo de las moras moradas***

Error absoluto

$$\boldsymbol{Error\ absoluto} = |\boldsymbol{Valor\ real} - \boldsymbol{Valor\ obtenido}|$$

$$Error\ absoluto = |172 - 148|$$

$$Error\ absoluto = 24$$

Error relativo

$$\boldsymbol{Error\ relativo} = \left|\frac{\boldsymbol{Valor\ teórico - Valor\ medido}}{\boldsymbol{Valor\ teórico}}\right| * \mathbf{100}$$

$$Error\ relativo = \left|\frac{172-148}{172}\right| * 100$$

$$Error\ relativo = 13{,}95\%$$

Tabla 31. Eficacia total del sistema.

Estado de maduración de la mora	Moras contadas manualmente	Moras contadas por el modelo automático	Error absoluto	Error relativo
Verde	308	224	84	27,27%
Roja	239	198	41	17,15%
Madura o morada	172	148	24	13,95%
Promedio				**19,45%**

En la Tabla 31 se muestra de manera detallada la eficiencia del sistema implementado para determinar el conteo de las moras verdes, rojas y maduras o moradas los resultados son los siguientes: el conteo de la mora verde es de una eficiencia de 72,73%, el conteo de mora roja es de 82,85% y por último la eficiencia de la mora madura es del 86,05%.

b. Confiabilidad del prototipo

Para calcular la confiabilidad del sistema se establece por medio del promedio de los errores relativos.

$$\boldsymbol{Confiabilidad_{sistema}} = \mathbf{100\%} - \boldsymbol{error_{promedio\ error\ relativo}} \qquad (\mathbf{5})$$

$$Confiabilidad_{sistema} = 100\% - 19{,}45\%$$

$$Confiabilidad_{sistema} = 80{,}55\%$$

La confiabilidad del sistema o del prototipo propuesto es del 80,55% por lo que garantiza un prototipo fiable, pero a la vez mejorable para futuras investigaciones, este resultado de confiabilidad es muy bueno ya que son datos en tiempo real, además otro factor a destacar es que la planificación de la trayectoria en el cultivo de mora ayudó a mejorar el tiempo de monitoreo y su respectivo conteo y clasificación, ahorrando tiempo en comparación a revisiones manuales.

3.3.7 Requisitos mínimos de hardware

En la Tabla 32 se observa los requerimientos mínimos de hardware para el sistema:

Tabla 32. Requisitos mínimos de hardware.

Categoría		**Especificación Mínima**
Procesamiento y Computación	**CPU**	Quad-core ARM® Cortex®-A57 MPCore processor
	GPU	128-core NVIDIA Maxwell architecture GPU
	Memoria RAM	4GB RAM
Almacenamiento	**Espacio en disco**	32 GB
	Tipo de disco	Tarjeta microSD v10
Periféricos y Sensores	**Cámara/Dron**	Compatible con DJI Mini 2 - Mini 4 Pro, Air, Mavic, Spark, Phantom, Inspire.
	Conectividad	WiFi en la misma red: Flask (interfaz) y Nginx (video por RTMP)
Requerimientos Energéticos	**Fuente de alimentación**	Micro-USB 5V 2A / DC power adapter 5V 4A
	Consumo de energía y refrigeración	NVIDIA Jetson Nano con disipador de calor y placa base de referencia

3.3.8 Requisitos mínimos de software

En la Tabla 33 se establece cuáles son los requerimientos mínimos de software para el sistema:

Tabla 33. Requerimientos mínimos de software

Componente	Software / Herramienta	Versión Mínima	Requerimientos Adicionales
Adquisición de imágenes	Rainbow for DJI	4.76	Compatible con DJI Mini 2
Procesamiento de imágenes	OpenCV, PIL, HSV	OpenCV 4.5+, PIL 8.0+	Soporte para imágenes de alta resolución
Modelo de IA	YOLOv8m	Ultralytics YOLOv8	Requiere PyTorch con aceleración GPU
Aceleración de hardware	CUDA	11.3+	Compatible con GPU NVIDIA
Base de datos	MongoDB	8.0+	Soporte para almacenamiento en la nube/local
Interfaz gráfica	Flask, JavaScript, HTML, CSS	Flask 3.0+	Compatible con MongoDB y Python
Lenguaje principal	Python	3.8+	Soporte para PyTorch, TensorFlow
Frameworks de IA	PyTorch, TensorFlow	PyTorch 1.10+, TensorFlow 2.6+	Soporte para GPU con CUDA
Bibliotecas adicionales	NumPy, SciPy, Matplotlib, pillow, sort, pymongo	Última estable	Dependencias para procesamiento y visualización
Sistema operativo	Linux (Ubuntu)	20.04 Focal	Compatibilidad con CUDA y GPU NVIDIA

3.3.9 Costo computacional

Los costos computacionales se basan básicamente en las consideraciones técnicas de la máquina, además del tiempo empleado a cada modelo entrenando, además el uso de Google Colab con GPU ya que ayuda de gran manera al entrenamiento en velocidad cuando son entrenamientos de modelos de aprendizaje profundo, debido a su capacidad de procesamiento, en la Tabla 34 se muestra lo mencionado anteriormente.

Tabla 34. Costos computacionales.

Parámetro		**Descripción**
MAQUINA	**CPU**	CPU Hp PAVILION
	RAM	13 GB
	GPU	128 Core-Maxwell
	TPU	Quad-core ARM A57 1.43GHz
	Memoria GPU	15 GB
	Sistema Operativo	Linux (Ubuntu)
	Almacenamiento	75.4 GB
Tipo de modelo		YOLOv8m

Tamaño del dataset	
Modelo 1	**Modelo 2**
3000 imágenes	1500 imágenes

Numero de épocas	
Modelo 1	**Modelo 2**
500 épocas	100 épocas

Tiempo de entrenamiento		
Modelo 1	**Modelo 2**	**Total**
600 min	90.74 min	**690.74 min / ~12 horas**
Uso de GPU	~12.4 GB/15GB	85-100%
Memoria RAM utilizada	~6.3/13 GB	45-70%

3.3.10 Presupuesto

El sistema de estimación de producción de cultivo de mora empleando visión artificial y machine learning tiene un presupuesto que se calcula considerando costos de diseño y de fabricación. En el presupuesto de diseño se considera el tiempo empleado en el desarrollo y el sueldo de un ingeniero en Telecomunicaciones en Ecuador que bordea los 858 dólares según el Ministerio de Trabajo, para esto se considera 5 días laborables en cada semana, 21 días laborables en el mes para determinar el salario.

$$\boldsymbol{Salario_{xdia}} = \frac{\boldsymbol{Salario_{xmes}}}{\boldsymbol{Dias\ laborables}} \qquad (\mathbf{6})$$

$$\boldsymbol{Salario_{xdia}} = \frac{858}{21}$$

$$\boldsymbol{Salario_{xdia}} = \$\ 40.86$$

Para calcular el salario por hora se toma en cuenta que la jornada laboral es de 8 horas al día:

$$\boldsymbol{Salario_{xhora}} = \frac{\boldsymbol{Salario_{xdía}}}{\boldsymbol{Horas\ laborables}} \quad (\mathbf{7})$$

$$\boldsymbol{Salario_{xhora}} = \frac{40.86\ dólares}{8\ horas}$$

$$\boldsymbol{Salario_{xhora}} = \$\ 5.11$$

El costo total del diseño se determina en un lapso de 4 meses trabajados, siendo un total de 120 horas, la hora de trabajo es de $ 5.36. Durante ese tiempo se realizó las siguientes actividades: investigación, recolección de datos para el dataset y su posterior entrenamiento, etiquetado de imágenes, entrenamiento del modelo y desarrollo de la interfaz gráfica.

$$\boldsymbol{Presupuesto_{diseño}} = \boldsymbol{Horas_{trabajo}} * \boldsymbol{Salario_{xhora}} \quad (\mathbf{8})$$

$$\boldsymbol{Presupuesto_{diseño}} = 120\ horas * \$\ 5.11$$

$$\boldsymbol{Presupuesto_{diseño}} = \$\ 613.20$$

En la Tabla 35 se detallan los costos de los materiales utilizados para la implementación del sistema:

Tabla 35. Presupuesto de implementación del sistema.

Ítem	Recursos	Cantidad	Precio Unitario	Precio Total
1	Dron DJI Mini2	1	$ 500	$ 500
2	Control RC para Dron DJI	1	$ 200	$ 200
3	Jetson Nano	1	$ 300	$ 300
4	Camara Steren 4K	1	$ 125	$ 125
5	Aplicación Rainbow for DJI Drones	1	$ 17	$ 17
6	Teclado	1	$ 5	$ 5
7	Monitor	1	$ 80	$ 80
8	Mouse	1	$ 10	$ 10
9	Adaptador HDMI a VGA	1	$ 10	$ 10
10	Tarjeta microSD de 64GB	1	$ 17	$ 17
			Subtotal	$ 1264
			IVA 15%	$ 189.60
			Total	$ 1453.60

El presupuesto total del sistema se determina por la siguiente ecuación:

$$\boldsymbol{Presupuesto_{total}} = \boldsymbol{Presupuesto_{diseño}} + \boldsymbol{Presupuesto_{implementación}} \quad (\mathbf{9})$$

$$\boldsymbol{Presupuesto_{total}} = \$\ 613.20 + \$\ 1453.60$$

$$\boldsymbol{Presupuesto_{total}} = \$\ 2066.80$$

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se ha llevado a cabo una investigación bibliográfica, se logró identificar y analizar las principales técnicas empleadas para la estimación de producción de mora, los métodos de estimación son generalmente por métodos tradicionales basados en observaciones de campos manuales y estimaciones empíricas, siendo útil en agricultores que no disponen de recursos económicos para emplear tecnologías en su producción, sin embargo esto presenta desventajas como la dependencia de gastos en agrónomos calificados, además de la susceptibilidad de los errores humanos. En este trabajo, se implementó un enfoque basado en sistemas tecnológicos y modelos de machine learning, lo que permitió la automatización del conteo de moras y la estimación de producción. Esta metodología mejora la eficiencia del proceso, reduce los tiempos de evaluación y minimiza los errores asociados a la intervención humana.

- Se ha documentado y analizado las técnicas disponibles en visión artificial y machine learning aplicables al sistema de estimación de producción de mora. Se implementó la técnica de drones con visión artificial, permitiendo un monitoreo eficiente de parcelas y la captura de datos a gran escala con alta precisión. Se utilizó un UAV multirrotor, específicamente el DJI Mini 2, seleccionado por su estabilidad, maniobrabilidad y accesibilidad económica en comparación con otras opciones más costosas. En el procesamiento de imágenes, se emplearon algoritmos de visión artificial basados en redes neuronales convolucionales (CNN) que permitió la detección, clasificación y conteo automatizado de frutos con gran precisión. La combinación de estas tecnologías optimizó la recolección de datos y redujo la dependencia de métodos manuales, mejorando la eficiencia y confiabilidad en la supervisión de cultivos.

- El modelo YOLOv8m demostró ser el algoritmo más eficiente en la detección y clasificación de moras, alcanzando métricas de precisión del 95%, recall del 91% y un mAP50-95 del 92%, superando significativamente a VGG16, que obtuvo una precisión del 86%. Además, el sistema procesó imágenes en tiempo real con un tiempo promedio de inferencia de 98.10 ms. La comparativa presentada en la Tabla 11 sobre vehículos arereos no tripulados (VANTs) del Capítulo 1, evidencia que el DJI Mini 2, equipado con una cámara 4K 1080p a 60 fps, mejoró la calidad de las imágenes y la detección de frutos en comparación con modelos anteriores de menor resolución. la implementación de segmentación en el espacio de color HSV optimizó la detección de frutos en distintos estados de madurez, mientras que el uso de SORT permitió un seguimiento preciso y un conteo eficiente de moras verdes, rojas y maduras. Estas mejoras ayudaron a la implementación del sistema en parcelas tecnificadas, dentro del marco de la agricultura de precisión.

- El dron DJI Mini 2, combinado con la aplicación Rainbow for DJI v4.76, facilitó la planificación de trayectorias optimizadas y la captura de imágenes de alta calidad en condiciones de campo. Esta integración redujo significativamente el tiempo y los recursos necesarios para una evaluación manual, garantizando una cobertura completa de las parcelas tecnificadas. Su estabilidad y calidad de captura contribuyeron al éxito del sistema en la detección precisa de frutos.

- La interfaz web desarrollada utilizando MongoDB 8.0.3 y Flask 3.1 permitió la gestión e interpretación eficiente de los datos generados por el sistema. A través de esta herramienta, se presentó información clara y organizada, como el conteo de moras clasificadas por estados de maduración y las estimaciones de producción en peso, facilitando la supervisión del proceso en tiempo real mediante video transmitido desde el UAV. Este desarrollo demuestra la importancia de las herramientas visuales intuitivas en la gestión agrícola automatizada.

- El sistema diseñado alcanzó una eficiencia del 80.55% en la detección, clasificación y conteo de moras, consolidándose como un prototipo funcional

que contribuye a mejorar la calidad y precisión de la producción agrícola. Su integración de visión artificial machine learning y planificación automatizada mediante UAV representa un avance significativo para la agricultura tecnificada en Ecuador, permitiendo una supervisión continua y decisiones basadas en datos confiables.

4.2 Recomendaciones

- Al no tener un dataset ya definido se debería crear uno desde cero por lo que es recomendable tener uno superior a 1500 imágenes para que el entrenamiento sea más eficiente y preciso en el modelo de estimación de producción mora. Además, las imágenes deben de ser tomadas desde diferentes ángulos y en horas en las cuales la iluminación del sol no afecte a la calidad de las imágenes.

- Se recomienda un desarrollo de un sistema que combine la visión artificial con datos relevantes para el cultivo de mora como la profundidad para conocer el tamaño especifico de la mora y las condiciones del suelo. Además, la integración de diagnósticos del estado del cultivo y predicción de problemas potenciales como estados climáticos, posibles sequias, heladas y otros factores que afecten de manera directa a la producción de mora en la cual se integren múltiples fuentes de información y así detectar con más precisión perdidas en la producción de mora.

- Se sugiere incorporar otros algoritmos de machine learning y técnicas de optimización que ayuden al sistema a adaptarse automáticamente a diferentes factores climáticos como vientos, lluvias, además de que la planificación de trayectorias mediante el UAV a medida que se recolectan los datos.

- En la planificación de trayectorias se recomienda utilizar sensores avanzados como LiDAR y cámaras multiespectrales, ya que el sistema empleado tiene margen de mejora en su precisión y eficacia de monitoreo del cultivo de mora, LiDIAR es una tecnología que se basa en un láser que mide las distancia entre el sensor y los objetos en su entorno lo cual garantiza vuelos seguros ya que opera de manera efectiva incluso en condiciones baja luz o nubosidad, y

optimiza las trayectorias y la cámara multiespectral ayuda de gran manera a distinguir de gran manera los frutos maduros, verdes, rojos y en dañados, mejorando la precisión en la estimación de producción.

- Para la interfaz gráfica se recomienda visualizarla a través de un bot, el cual sea intuitivo y personalizado para ser usado por los agricultores o técnicos, que permita no solo visualizar los datos del conteo de mora y su estimación, sino que también se personalice parámetros como rangos de madurez del fruto, áreas de interés en el cultivo y tiempos de monitoreo en filas o plantas en específico para que las plantas tengan un desarrollo más eficiente en su producción.

REFERENCIAS BIBLIOGRÁFICAS

[1] Ministerio de Agricultura y Desarrollo Rural, «Mingagricultura,» 2019. [En línea]. Available: https://sioc.minagricultura.gov.co/Mora/Documentos/2019-12-30%20Cifras%20Sectoriales.pdf. [Último acceso: 15 Junio 2024].

[2] Ministerio de Agricultura y Ganaderia, «Boletín Situacional Mora 2021,» 2022.

[3] O. E. Apolo Apolo, M. Pérez Ruiz, G. Egea y J. Martínez Guanter, «Estimación de producción en cítricos usando técnicas de aprendizaje automático,» *X Congreso Ibérico de Agroingeniería,* pp. 44-51, 2019.

[4] S. A. Shah, «Application of drone surveillance for advance agriculture monitoring by Android application using Convolution Neural Network,» *Agricultura,* vol. 13, nº 7, 29 junio 2023.

[5] L. E. Sánchez Palacios, F. R. Martínez Alcivar, S. T. Torres Sánchez, A. C. Lascano Montes y G. N. Terán Guajala, «Agricultura de Precisión en El Ecuador,» *Ciencia Latina Revista Cientifica Multidisciplinar,* vol. 8, nº 1, pp. 1532-1542, 2024.

[6] C. G. Feicán Mejía, H. H. Hugo Fernando , M. S. Aníbal Arturo y V. D. Pablo Francisco , Guía para facilitar el aprendizaje en el manejo integrado del cultivo de mora (Rubus glaucus Benth), Cuenca: Instituto Nacional de Investigaciones Agropecuarias, 2020.

[7] Ministerio de Agricultura y Ganaderia, «FLIPHTML5,» Julio 2024. [En línea]. Available: https://fliphtml5.com/ijia/ngol/Bolet%C3%ADn_Situacional_Mora_2023/. [Último acceso: 15 Septiembre 2024].

[8] M. Iza, P. Viteri, M. Hinojosa, A. Martínez, A. Sotomayor y W. Viera, «Diferenciación morfológica, fenológica y pomológica de cultivares comerciales de mora (Rubus glaucus Benth.),» *Enfoque UTE*, vol. 11, nº 2, pp. 47-57, 2020.

[9] INEN, «NORMA TECNICA ECUATORIANA,» de *FRUTAS FRESCAS. MORA. REQUISITOS*, Quito, 2016, p. 15.

[10] M. M. Bolaños Benavides, W. A. Cardona, M. C. García Muñoz, Y. A. Zapata Narváez, C. R. Beltrán Acosta y D. A. López Melo, Mora (Rubus glaucus Benth.) Manual de recomendaciones técnicas para su cultivo en el departamento de Cundinamarca, Bogota, 2020.

[11] «Artificial Intelligence in Smart Agriculture: Applications and Challenges,» *Current Applied Science and Technology,* vol. Vol. 24, nº 2, 9 Noviembre 2023.

[12] C. R. Monroy Alfaro, «EL LENGUAJE PYTHON Y SU POTENCIAL EN EL DESARROLLO DE SOFTWARE DE INTELIGENCIA ARTIFICIAL,» 2020.

[13] G. Sidorov, Métodos modernos de Inteligencia Artificial, Mexico D. F, 2011.

[14] G. J. Chacaltana Buleje y A. S. Ramírez Fernández, Lenguajes de Programación C y C++ - Inteligencia Artificial, Universidad Ricardo Palma: Lima, 2019.

[15] A. Ramos Martín y J. Ramos Martín, Entornos de desarrollo, Madrid: Editorial Garceta, 2014.

[16] INEC, «Información ambiental y tecnificación agropecuaria - Módulo métodos de producción y ambiente.,» ESPAC 2023, 2023.

[17] INFAIMON, «SOLUCIONES DE VISIÓN ARTIFICIAL PARA EL SECTOR,» INFAIMON, 2021.

[18] C. Platero Dueñas, Apuntes de Vision Artificial, Madrid: Universidad Politecnica de Madrid, 2008.

[19] P. M. García Arias, J. I. Avilés Monroy, V. Mendoza Morán y J. R. Hechavarría Hernández, «Drones para la agricultura de precisión, un enfoque al cultivo de la Pitahaya,» *ResearchGate*, Noviembre 2022.

[20] H. Pérez León , «Generación y seguimiento de trayectorias para un vehículo aéreo multi-rotor,» Dep. de Ingeniería de Sistemas y Automática Universidad de Sevilla, Sevilla, 2018.

[21] DJI, [En línea]. Available: https://www.dji.com/global/support/product/mini-2. [Último acceso: 15 Octubre 2024].

[22] SMART OPEN LAB, [En línea]. Available: https://smartopenlab.com/maquina/drone-dji-tello-edu/. [Último acceso: 15 Octubre 2024].

[23] DJI, [En línea]. Available: https://www.dji.com/support/product/phantom-3-adv. [Último acceso: 15 Octubre 2024].

[24] R. Miranda Colorado, R. A. Garrido Moctezuma, L. T. Aguilar Bustos y J. E. Herrero Brito, Drones. Modelado y control de cuadricópteros, Marcombo, 2020.

[25] Geoinnova, 26 05 2021. [En línea]. Available: https://geoinnova.org/blog-territorio/vuelo-fotogrametrico/. [Último acceso: 23 11 2024].

[26] DRONE PRIX BLOG, 10 09 2024. [En línea]. Available: https://blog.droneprix.es/guia-dp-planificar-una-mision-de-fotogrametria-con-drones/. [Último acceso: 23 11 2024].

[27] Dronelink, [En línea]. Available: https://www.dronelink.com/. [Último acceso: 23 11 2024].

[28] Softonic, [En línea]. Available: https://rainbow-for-dji-mavic-phantom-inspire-spark.softonic.com/android. [Último acceso: 23 11 2024].

[29] Drone Harmony, [En línea]. Available: https://www.droneharmony.com/. [Último acceso: 23 11 2024].

[30] D. Rodríguez, «Regresión de Vectores de Soporte (SVR, Support Vector Regression),» 17 Diciembre 2021.

[31] Servicios Web de Amazon, [En línea]. Available: https://aws.amazon.com/es/what-is/monte-carlo-simulation/. [Último acceso: 22 Noviembre 2024].

[32] IBM, [En línea]. Available: https://www.ibm.com/es-es/topics/monte-carlo-simulation. [Último acceso: 23 Noviembre 2024].

[33] Emprendedores, 15 Julio 2024. [En línea]. Available: https://emprendedores.es/gestion/metodo-montecarlo-como-aplicarlo/#:~:text=por%20las%20empresas.-,Desventajas%20del%20m%C3%A9todo%20Montecarlo,ya%20que%20podr%C3%ADa%20tener%20fallos.. [Último acceso: 23 Noviembre 2024].

[34] «Fast R-CNN,» [En línea]. Available: https://github.com/rbgirshick/. [Último acceso: 22 Septiembre 2024].

[35] L. Alegsa, «ALEGSA,» 1 Julio 2023. [En línea]. Available: https://www.alegsa.com.ar/Dic/microcomputadora.php#h0. [Último acceso: 30 Agosto 2024].

[36] NVIDIA . DEVELOPER, [En línea]. Available: https://developer.nvidia.com/embedded/jetson-nano. [Último acceso: 31 Octubre 2024].

[37] Raspberry Pi, [En línea]. Available: https://www.raspberrypi.org/. [Último acceso: 31 Octubre 2024].

[38] INTEL.LA, [En línea]. Available: https://www.intel.la/content/www/xl/es/homepage.html. [Último acceso: 31 Octubre 2024].

[39] B. V, «Hostinger,» 3 Mayo 2023. [En línea]. Available: https://www.hostinger.es/tutoriales/sgbd. [Último acceso: 20 Octubre 2024].

[40] SQLite, «SQLite,» [En línea]. Available: https://www.sqlite.org/docs.html. [Último acceso: 22 Octubre 2024].

[41] PostgreSQL, «PostgreSQL,» [En línea]. Available: https://www.postgresql.org/docs/15/. [Último acceso: 22 Octubre 2024].

[42] MongoDB, «MongoDB,» [En línea]. Available: https://www.mongodb.com/es. [Último acceso: 22 Octubre 2024].

ANEXOS

Anexo A. UAV DJI Mini2

En la Figura A1 se muestra las especificaciones técnicas del UAV DJI Mini2.

	DJI MINI 2
DIMENSIONES Y PESO	245 × 289 × 56 mm 243 gramos
FRECUENCIA DE FUNCIONAMIENTO	2.400 - 2.4835 GHz; 5.725 - 5.850 GHz
BATERÍA	2.250 mAh - 20 min aprox de vuelo - 2 horas, 10 min aprox de carga
PESO CONTROL REMOTO	346 g
DISTANCIA DE TRANSMISIÓN	10 km (FCC) - 6km (CE/SRRC/MIC)
ALTURA MÁXIMA DE VUELO	120m (aunque depende de la legislación de cada zona)
VELOCIDAD MÁXIMA DE VUELO	16 m/s (modo sport)
MODOS DE VUELO	Cine, normal y sport
SENSOR DE LA CÁMARA	CMOS de 1/2.3" de 12 MP
OBJETIVO	Campo de visión 83° Apertura f/2.8
TAMAÑO DE LA IMAGEN	4K
TAMAÑO DE VÍDEO	4K: 3840×2160 a 24/25/30 fps 2.7K: 2720×1530 a 24/25/30/48/50/60 fps FHD: 1920×1080 a 24/25/30/48/50/60 fps
APP	DJI Fly (iOS y Android)

Figura A1. Especificaciones técnicas DJI Mini 2

Anexo B. NVIDIA Jetson Nano

En la Figura B1 se muestra las especificaciones técnicas del microordenador.

Jetson Nano module

- 128-core NVIDIA Maxwell GPU
- Quad-core ARM® A57 CPU
- 4 GB 64-bit LPDDR4
- 16 GB eMMC 5.1
- 10/100/1000BASE-T Ethernet

Power

- Voltage Input: 5 V
- Module Power: 5 W–10 W

Environment

- Operating Temperature: -25 C to 80 C*
- Storage Temperature: -25 C to 80 C
- Humidity: 85% RH, 85°C (non-operational)
- Vibration: Sinusoidal 5 G RMS 10 to 500 Hz, random 2.88 G RMS, 5 to 500 Hz (non-operational)
- Shock: 140 G, half sine 2 ms duration (non-operational)

NVIDIA JETSON NANO MODULE
TECHNICAL SPECIFICATIONS

GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	16 GB eMMC 5.1
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
CSI	12 (3x4 or 4x2) lanes MIPI CSI-2 D-PHY 1.1
Connectivity	Gigabit Ethernet
Display	HDMI 2.0, eDP 1.4, DP 1.2 (two simultaneously)
PCIe	1x1/2/4 PCIe Gen2
USB	1x USB 3.0, 3x USB 2.0
Others	I²C, I²S, SPI, UART, SD/SDIO, GPIO
Mechanical	69.6 mm x 45 mm 260-pin SODIMM Connector

Figura B1. Especificaciones técnicas de la Jetson Nano.

Anexo C. Interfaz Web.

En la Figura C1 se muestra la interfaz gráfica con su autenticación de usuario.

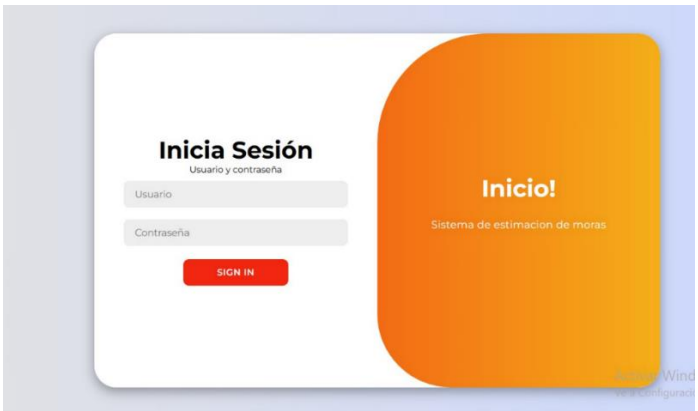

Figura C1. Inicio de la interfaz gráfica.

Anexo D. Hardware Adquirido.

En la Figura D1 se muestra el don DJI Mini 2.

Figura D1. Dron DJI Mini 2.

En la Figura D2 se muestra la Jetson Nano que se va a ocupar en la investigación.

Figura D2. NVIDIA Jetson Nano.

Anexo E. Cultivo de mora.

En la Figura E1 se muestra cómo está el cultivo de mora.

Figura E1. Cultivo de mora.

En la Figura E2 se muestra a los investigadores con el ingeniero agrónomo.

Figura E2. Investigadores en el cultivo de mora.

En la Figura E3 se muestra la inspección manual de las moras por el ingeniero agrónomo.

Figura E3. Inspección manual de las moras.

En la Figura E4 se muestra las pruebas de vuelo con dron.

Figura E4. Primeras pruebas de vuelo.

Anexo F.Código del sistema de estimación.

```
from flask import Flask, Response, send_file, jsonify, render_template, request  # Importación de Flask y funciones auxiliares para manejar respuestas HTTP

import cv2  # OpenCV para manipulación de imágenes y video

import os  # Módulo para interactuar con el sistema operativo

import numpy as np  # Biblioteca para cálculos numéricos

from ultralytics import YOLO  # Importación del modelo YOLO para detección de objetos

from sort import Sort  # Importación del algoritmo de seguimiento de objetos Sort

import pymongo  # Biblioteca para interactuar con bases de datos MongoDB

# Inicialización de la aplicación Flask

app = Flask(__name__, template_folder=os.path.dirname(__file__))

# Configuración de la conexión a MongoDB

client = pymongo.MongoClient("mongodb://localhost:27017/")  # Conexión al servidor MongoDB local

db = client["detecciones_db"]  # Selección de la base de datos

collection = db["detecciones2"]  # Selección de la colección donde se almacenan las detecciones

# Inicialización de variables globales

camera = None  # Variable para almacenar la cámara

video_running = False  # Bandera para controlar el estado del video

txt-clase_0_ids = set()  # Conjunto para almacenar IDs de objetos de clase 0

txt-clase_1_ids = set()  # Conjunto para almacenar IDs de objetos de clase 1

txt-clase_2_ids = set()  # Conjunto para almacenar IDs de objetos de clase 2

# Inicialización del modelo YOLO y los rastreadores

model = YOLO("best1_2.pt")  # Carga del modelo YOLO entrenado

tracker = Sort()  # Inicialización del rastreador para la clase 0

tracker2 = Sort()  # Inicialización del rastreador para la clase 1

tracker3 = Sort()  # Inicialización del rastreador para la clase 2
```

```
# Función para almacenar detecciones en MongoDB
def guardar_detecciones():
    cont_rojas = len(clase_0_ids)  # Conteo de objetos clase 0
    cont_verdes = len(clase_1_ids)  # Conteo de objetos clase 1
    cont_moradas = len(clase_2_ids)  # Conteo de objetos clase 2
    totales = cont_rojas + cont_verdes + cont_moradas  # Suma total de objetos detectados

    # Creación del documento a almacenar en MongoDB
    documento = {
        "cantidad_rojas": cont_rojas,
        "cantidad_verdes": cont_verdes,
        "cantidad_moradas": cont_moradas,
        "totales": totales
    }
    collection.insert_one(documento)  # Inserción del documento en la base de datos
    print(f"Guardado: {documento}")  # Impresión del documento guardado

# Generador de frames para el flujo de video
def gen_frames():
    global camera, video_running
    while video_running:
        success, frame = camera.read()  # Captura un frame del video
        if not success:
            break
        else:
```

```
        # Detección de objetos usando YOLO
        results = model(frame, stream=True, classes=[0, 1, 2])

        # Procesamiento de detecciones
        for res in results:
            filtered_indices = np.where((res.boxes.conf.cpu().numpy() > 0.70) & (res.boxes.cls.cpu().numpy() == 0))[0]
            filtered_indices2 = np.where((res.boxes.conf.cpu().numpy() > 0.70) & (res.boxes.cls.cpu().numpy() == 1))[0]
            filtered_indices3 = np.where((res.boxes.conf.cpu().numpy() > 0.30) & (res.boxes.cls.cpu().numpy() == 2))[0]

            # Extraer coordenadas de las cajas
            boxes = res.boxes.xyxy.cpu().numpy()[filtered_indices].astype(int)
            boxes2 = res.boxes.xyxy.cpu().numpy()[filtered_indices2].astype(int)
            boxes3 = res.boxes.xyxy.cpu().numpy()[filtered_indices3].astype(int)

            # Actualizar rastreadores
            tracks = tracker.update(boxes)
            tracks2 = tracker2.update(boxes2)
            tracks3 = tracker3.update(boxes3)

            # Dibujar rectángulos y etiquetas
            for (xmin, ymin, xmax, ymax, track_id) in tracks:
                xmin, ymin, xmax, ymax = map(int, [xmin, ymin, xmax, ymax])
                cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 0, 255), 3)
                cv2.putText(frame, f"Mora {track_id} - Roja", (xmin, ymin - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 4)
```

```
            clase_0_ids.add(track_id)


        for (xmin2, ymin2, xmax2, ymax2, track_id2) in tracks2:
            xmin2, ymin2, xmax2, ymax2 = map(int, [xmin2, ymin2, xmax2, ymax2])
            cv2.rectangle(frame, (xmin2, ymin2), (xmax2, ymax2), (277, 131, 277), 3)
            cv2.putText(frame, f"Mora {track_id2} - Verde", (xmin2, ymin2 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (277, 131, 277), 4)
            clase_1_ids.add(track_id2)


        for (xmin3, ymin3, xmax3, ymax3, track_id3) in tracks3:
            xmin3, ymin3, xmax3, ymax3 = map(int, [xmin3, ymin3, xmax3, ymax3])
            cv2.rectangle(frame, (xmin3, ymin3), (xmax3, ymax3), (255, 0, 255), 3)
            cv2.putText(frame, f"Mora {track_id3} - Morada", (xmin3, ymin3 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 255), 4)
            clase_2_ids.add(track_id3)


        # Codificar la imagen en formato JPEG
        _, buffer = cv2.imencode('.jpg', frame)
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + buffer.tobytes() + b'\r\n')


# Ruta para iniciar el video
@app.route('/start_video', methods=['POST'])
def start_video():
    global camera, video_running
    if not video_running:
        camera = cv2.VideoCapture("VID_20241211_180521.mp4")
```

```
        if not camera.isOpened():
            return jsonify({"status": "Error al abrir la cámara"}), 500
        video_running = True
    return jsonify({"status": "Video iniciado"})

# Ruta para detener el video
@app.route('/stop_video', methods=['POST'])
def stop_video():
    global camera, video_running
    video_running = False
    guardar_detecciones()
    if camera:
        camera.release()
        camera = None
    return jsonify({"status": "Video detenido"})

# Ruta para enviar el flujo de video
@app.route('/video_feed')
def video_feed():
    if not video_running or not camera:
        return Response("Video no activo", status=503)
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

# Servir archivos HTML y CSS

# Ruta para servir el archivo index.html
@app.route('/')
def index():
```

```
    return send_file(os.path.join(os.path.dirname(__file__), 'index.html'))

# Ruta para servir el archivo de estilos CSS

@app.route('/styles.css')

def styles():

    return send_file(os.path.join(os.path.dirname(__file__), 'styles.css'))

# Ruta para mostrar la tabla de documentos almacenados en la base de datos

@app.route("/tabla")

def tabla():

    documentos = collection.find()  # Recupera los documentos desde la base de datos

    return render_template("tabla.html", documentos=documentos)  # Renderiza la plantilla con los datos

# Función para realizar la simulación de Monte Carlo

def montecarlo_simulation(prob_plaga, prob_helada, cantidad_rojas, cantidad_verdes, cantidad_moradas, totales, num_simulaciones=10000):

    #Realiza una simulación de Monte Carlo para predecir el impacto de plaga y helada en la cosecha.

    :param prob_plaga: Probabilidad de que ocurra una plaga (0 a 1)

    :param prob_helada: Probabilidad de que ocurra una helada (0 a 1)

    :param cantidad_rojas: Cantidad de frutas rojas

    :param cantidad_verdes: Cantidad de frutas verdes

    :param cantidad_moradas: Cantidad de frutas moradas

    :param totales: Total de frutas (rojas + verdes + moradas)

    :param num_simulaciones: Número de simulaciones a realizar (por defecto 10,000)

    :return: Un valor que representa el porcentaje promedio de frutas afectadas por plaga y helada

    """

    total_frutas_afectadas = 0  # Inicializa el contador de frutas afectadas


    for _ in range(num_simulaciones):
```

```
    # Calcular las frutas afectadas por plaga basado en probabilidad
    afectadas_por_plaga = (prob_plaga * 0.30 * cantidad_rojas) + (prob_plaga * 0.10 * cantidad_moradas)

    # Calcular las frutas afectadas por helada basado en probabilidad
    afectadas_por_helada = prob_helada * 0.20 * cantidad_verdes

    # Sumar las frutas afectadas por plaga y helada
    frutas_afectadas = afectadas_por_plaga + afectadas_por_helada
    total_frutas_afectadas += frutas_afectadas

  # Calcular el promedio de frutas afectadas
  promedio_afectadas = total_frutas_afectadas / num_simulaciones

  # Calcular el porcentaje de frutas afectadas en relación al total de frutas
  porcentaje_afectadas = (promedio_afectadas / totales) * 100

  return round(porcentaje_afectadas, 2)  # Redondear a 2 decimales para obtener un valor en porcentaje

# Ruta para cargar la simulación de Monte Carlo
@app.route("/montecarlo", methods=['GET', 'POST'])
def montecarlo():
  if request.method == 'POST':
    # Obtener los valores enviados desde el formulario de la página web
    prob_plaga = float(request.form['prob_plaga'])
    prob_helada = float(request.form['prob_helada'])
```

```
    # Leer los últimos documentos de la colección de MongoDB
    documento = collection.find().sort([('_id', -1)]).limit(1)  # Obtener el último documento almacenado
    for doc in documento:
      cantidad_rojas = doc.get('cantidad_rojas', 0)
      cantidad_verdes = doc.get('cantidad_verdes', 0)
      cantidad_moradas = doc.get('cantidad_moradas', 0)
      cantidad_brotes = doc.get('cantidad_brotes', 0)
      totales = doc.get('totales', 0)

    # Realizar la simulación de Monte Carlo con los valores obtenidos
    prediccion = montecarlo_simulation(prob_plaga, prob_helada, cantidad_rojas, cantidad_verdes, cantidad_moradas, totales)

    # Renderizar los resultados en la misma página de Monte Carlo
    return render_template("montecarlo.html", prediccion=prediccion, documento=doc)

  # Si es un GET, simplemente mostrar la página sin cálculos previos
  return render_template("montecarlo.html")
# Ruta para sumar las filas de una tabla recibida en formato JSON
@app.route('/sumar_filas', methods=['POST'])
def sumar_filas():
  datos = request.json.get('datos', [])  # Obtiene los datos enviados en formato JSON
  if not datos:
    return jsonify({"error": "No se enviaron datos"}), 400  # Retorna un error si no se enviaron datos
  # Inicializar un arreglo con ceros para las sumas
```

```
    sumas = [0] * len(datos[0])

    # Sumar cada columna

    for fila in datos:

        for i, valor in enumerate(fila):

            sumas[i] += valor

    return jsonify({"sumas": sumas})  # Retorna la suma de cada columna en formato JSON

# Ejecutar la aplicación Flask si el script es ejecutado directamente

if __name__ == "__main__":
    app.run(debug=True, port=5000)  # Inicia el servidor en el puerto 5000 con depuración habilitada
```

Anexo G. Abreviaturas de palabras de la tesis.

Abreviatura	Definición
API	Interfaz de programación de aplicaciones
ARM	Máquina RISC avanzada
AWS	Amazon Web Services
CCD	Dispositivo de carga acoplada
CCIR	Sistema de transmisión de televisión analógica
CMD	Control por Modos Deslizantes
CNN	Redes Neuronales Convolucionales
Concat	Junto
CSI	Computadoras y estructuras, Inc.
CUDA	Arquitectura Unificada de Dispositivos de Cómputo
CuDNN	NVIDIA CUDA Deep Neural Network (Red neuronal profunda NVIDIA CUDA)
DL	Deep Leraning
Dlib	Biblioteca de código abierto
DJI	Da-Jiang Innovations (Grandes Innovaciones Fronterizas)
EFS	Entidades Fiscalizadoras Superiores
FOV	Campo de visión
FPS	Fotogramas por Segundo
FPV	Visión en Primera Persona
GPL	Licencia Pública General
GPS	Sistema de Posicionamiento Global
GPU	Unidad de procesamiento de gráficos
Ha	Hectáreas
HDMI	High Definition Multimedia Interface (Interfaz Multimedia de Alta Definición)

HVS	Hue, Saturation, Value (Matiz, Saturación, Valor)
IA	Inteligencia Artificial
IAM	Gestión de Identidad y Acceso
INIAP	Instituto Nacional de Investigaciones Agropecuarias
IoU	Solapamiento de objetos
JIT	Just-in-Time (Justo a tiempo)
JPEG	Joint Photographic Experts Group (Grupo Mixto de Expertos en Fotografía)
JSON	Notación de objetos JavaScript
LiDAR	Detección de luz y rango
MDE	Modelo digital del terreno
MQL	Cliente potencial calificado por marketing
NoSQL	no solo SQL
NTSC	Comité Nacional de Sistema de Televisión
OpenCV	Visión Computacional Abierta
ORDBMS	Sistema de gestión de bases de datos relacional extendida
OTC	On-The-Go (En marcha)
PAL	Línea de fase alternada
PD	Proporcional-derivativo
PI	Proporcional-integral
PID	Proporcional-Integral-Derivativo
RBAC	Control de Acceso Basado en Roles
RCL	Regulador Cuadrático Lineal
RDBMS	Sistema de gestión de bases de datos relacional
RGB	Rojo, Verde y Azul
RNN	Redes Neuronales Recurrentes
ROI	Extracción de la región de interés
RTK	Cinemática en tiempo real
RTMP	Protocolo de mensajería en tiempo real
SDK	Kit de Desarrollo de Software
SGBD	Sistema de Gestión de Base de Datos
SGBDR	Sistema de Gestión de Bases de Datos Relacionales
SIG	Sistemas de Información Geográfica
SORT	Ordenar o categorizar
SPPF	Agrupación de pirámides espaciales: rápida
SQ	Consulta Estructurada
SQL	Lenguaje de Consulta Estructurada
SVM	Máquinas de Soporte Vectorial
T	Toneladas
TensorFlow	Biblioteca de software de código abierto que permite crear modelos de aprendizaje automático
TPU	Unidades de procesamiento tensorial
UAV	Vehículo aéreo no tripulado
UI	Interfaz de usuario
USB	Bus Universal en serie
VANT	Vehículo aéreo no tripulado
VGA	Video Graphics Array (Matriz de gráficos de video)
VGG	Visual Geometry Group (Grupo de Geometría Visual)

Weka	Entorno para análisis del conocimiento de la Universidad de Waikato
WSG84	Sistema Geodésico Mundial 1984
Yolo	You Only Look Once (Solo mira una vez)
YOLOv8	You Only Look Once versiones 8 (Solo mira una vez versión 8)