IDENTIFICACIÓN DE SURCOS, SEGUIMIENTO Y MAPEO DE CULTIVOS APOYADO EN UNA PLATAFORMA AÉREA

DIEGO ALEJANDRO PRIETO PEDRAZA

ADRIANA RIVEROS GUEVARA

LEONARDO ENRIQUE SOLAQUE GUZMÁN



UNIVERSIDAD MILITAR NUEVA GRANADA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA MECATRÓNICA

BOGOTÁ D.C., SEPTIEMBRE DE 2021

IDENTIFICACIÓN DE SURCOS, SEGUIMIENTO Y MAPEO DE CULTIVOS APOYADO EN UNA PLATAFORMA AÉREA

DIEGO ALEJANDRO PRIETO PEDRAZA

ADRIANA RIVEROS GUEVARA

LEONARDO ENRIQUE SOLAQUE GUZMÁN



UNIVERSIDAD MILITAR NUEVA GRANADA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA MECATRÓNICA

BOGOTÁ D.C., SEPTIEMBRE DE 2021

Nota aceptación

_____
_____
_____
_____

_____
Firma de tutor

_____
Firma de co-tutor

_____
Firma de jurado 1

_____
Firma de jurado 2

Bogotá D. C., septiembre de 2021

Dedicatoria

Quiero dedicar esta tesis a Dios, quien me ha brindado todas las oportunidades que me han permitido llegar hasta donde estoy.

A mi familia, a mis padres por apoyarme durante toda mi vida y en especial en esta etapa tan importante y llena de retos. A mis abuelos, tíos y hermanos que han estado al pendiente de lo que he necesitado durante mi carrera, en especial a mi tía Rosmery porque gracias a ella me fue posible iniciar con mis estudios universitarios.

Agradecimientos

Al grupo de Investigación y Desarrollo de Aplicaciones Móviles (GIDAM) y especialmente a mis tutores, Ing. Adriana Riveros M.Sc. y el Ing. Leonardo Solaque PhD. Muchas gracias por darme la oportunidad de tomar este proyecto y por sus constantes observaciones y sugerencias, su acompañamiento fue esencial para la mejora continua y la culminación de esta investigación. Agradezco también su ayuda durante toda la gestión que se requirió para presentar mi trabajo de grado.

Al ingeniero Guillermo Sánchez por su orientación e indicaciones sobre los elementos requeridos al comienzo de este trabajo.

Agradezco a cada uno de mis amigos y compañeros de carrera que me brindaron su apoyo en el momento en que lo necesitaba, me alegra saber que habrá profesionales como ustedes siempre dispuestos a hacer las cosas con excelencia, con cariño y de la mejor manera. Gracias a ustedes la universidad no sólo fue un espacio académico sino un lugar donde pude vivir experiencias muy gratas durante los últimos años.

Gracias al docente Ing. Mauricio Mauledoux Ph.D. por instruirme durante sus clases de control y en el proyecto de Iniciación Científica que desarrollamos junto a mis compañeros Ing. Camilo Guachetá y la Ing. Margarita Hurtado. A la profesora Ing. Olga Ramos Ph.D. por la calidad de su enseñanza y la formación profesional que me inculcó. Al profesor Ing. Ricardo Castillo Ph.D. porque gracias a su clase de Inteligencia Artificial acogí un gran interés en esta temática.

Resumen

Dada la importancia que representa el sector agrícola para Colombia y los distintos avances e investigaciones que se han realizado con respecto a la agricultura de precisión, para el presente proyecto se decidió hacer uso de tecnologías que permitieran realizar la exploración de un cultivo simulado, apoyado en un vehículo aéreo.

En este trabajo se emplearon técnicas de procesamiento de imagen (Operaciones morfológicas, segmentación por color, filtro de promedio) y algoritmos de inteligencia artificial (redes convolucionales, YOLOv4) para identificar líneas de cultivo y posibles obstáculos que se puedan encontrar en un cultivo simulado, estas imágenes fueron obtenidas a partir de una cámara embarcada en un dron. A medida que se realiza este proceso, se genera una trayectoria la cual es seguida por el dron para posteriormente continuar con la toma de imágenes y el desplazamiento hasta llegar al final de cultivo. Los datos de ubicaciones de los surcos, trayectoria y obstáculos son guardados para ser procesados y generar un mapa.

Como resultado se obtuvo una precisión de validación de 0.87 en la detección de elementos del cultivo y una precisión en la detección de obstáculos de 0.91. La trayectoria generada mediante la identificación de surcos y el uso del algoritmo A* fue óptima para evadir los obstáculos presentes en el cultivo. Finalmente, las gráficas de error en el desplazamiento del vehículo aéreo con respecto a las posiciones deseadas tienden a un valor de 0. Estos resultados permitirán brindar información del cultivo a la plataforma terrestre Ceres_Agrobot.

Contenido

1. Introducción 9
2. Planteamiento del problema 11
3. Objetivos 12
3.1. Objetivo General 12
3.2. Objetivos Específicos 12
4. Estado del arte 13
4.1. Aplicación de los drones en investigación 14
4.2. Algoritmos usados en conjunto con UAVs 15
4.3. SLAM 15
4.4. Procesamiento de imagen 18
4.4.1. Incidencia de las condiciones de iluminación 21
5. De la simulación y sus elementos 27
5.2. El ambiente de simulación 28
5.3. El cultivo en Gazebo: 28
5.4. El hexacóptero 29
5.5. Integración con la cámara 29
5.6. El desplazamiento de la aeronave 30
6. De la detección de cultivos 32
6.1. Segmentación de grises por valor umbral 32
6.2. Segmentación de colores - Matlab 34
6.3. Operaciones morfológicas 36
6.3.1. Erosión 36
6.3.2. Dilatación 38
6.4. Filtro de promedio 40
6.5. Segmentación por valor umbral II 41
7. Las texturas y cultivos con bolsas de plástico 43
7.1. Segmentación de textura por filtro de Gabor 43
7.2. Segmentación de textura por filtro de entropía 44
7.3. Matriz de co-ocurrencia GLCM 45
7.4. División de imagen y clasificación de texturas 48
7.5. Red Neuronal para propiedades de imagen – matriz de co-ocurrencia 49
7.6. Redes convolucionales: 51

7.7. Resultado final del proceso de detección de cultivos desde Gazebo: 54
8. Delimitación del camino de cultivo 56
8.2. Detección de límites de cultivo 58
8.3. Detección de centros de surco a partir de límites de cultivo 60
8.4. El problema de la perspectiva 61
8.4.1. Mapeo inverso de perspectiva 61
9. Detección de objetos 63
9.1. Detección de objetos con Red Convolucional 63
9.2. Detección de objetos con Modelo YOLO 64
9.2.1. Carga y etiqueta de las clases 65
9.2.2. Dataset de entrenamiento 66
9.2.3. Entrenamiento del algoritmo 68
9.2.4. Ejecución del algoritmo 68
9.2.5. Detección realizada por el algoritmo 70
10. Evasión de obstáculos 71
11. Integración del simulador con el proceso de detección de cultivos 77
11.2. Equivalencia Píxel – Distancia 78
12. Generación de mapa 81
13. Resultados 82
14. Conclusiones y trabajos futuros 87
15. Bibliografía 89

Tabla de Figuras

Figura 1. Ceres_Agrobot, Robot dedicado a labores de agricultura de precisión. 13
Figura 2. Dron Matrice 600. Hexacóptero diseñada para aplicaciones industriales y fotografía aérea profesional. 14
Figura 3. Ambiente de simulación del cultivo en Gazebo..... 28
Figura 4. Hexacóptero Asctec Firefly usado en el proyecto. 29
Figura 5. Diagrama de relación entre nodos y tópicos de la simulación. 31
Figura 6. Cultivo en escala de grises. 33
Figura 7. Segmentación de imagen Cultivo en escala de grises a partir de valor umbral. 34
Figura 8. Toma original de cultivos en la Universidad Militar Nueva Granada. 35
Figura 9. Máscara obtenida de la segmentación de colores. 35
Figura 10. Regiones de verde en el cultivo a partir de máscara de segmentación por color. 36
Figura 11. Imagen y elemento estructural utilizados como ejemplo para realizar operaciones morfológicas. 37
Figura 12. Operación morfológica de erosión aplicada a imagen.. 37
Figura 13. Erosión de imagen usando rectángulo como elemento estructural. 38
Figura 14. Operación morfológica de dilatación aplicada a imagen. 39
Figura 15. Dilatación de imagen usando rectángulo como elemento estructural. 40
Figura 16. Filtro de promediado de imagen. 41
Figura 17. Segmentación posterior a filtro de promedio. 42
Figura 18. Filtro de Gabor aplicado en imagen para segmentación texturas de pasto y bolsa.. 44
Figura 19. Filtro de entropía aplicado en imagen para segmentación texturas de pasto y bolsa. ……45
Figura 20. Imagen para generar matriz de co-ocurrencia. 46
Figura 21. Matriz de co-ocurrencia. 46
Figura 22. Matríz de co-ocurrencia simétrica 46
Figura 23. Matriz de co-ocurrencia normalizada. 47
Figura 24. Gráfica de propiedades obtenidas a partir de la matriz de Co-Ocurrencia..... 49
Figura 25. División de imagen en celdas para uso de red convolucional..... 53
Figura 26. Segmentación de imagen luego de pasar por detección de elementos mediante red convolucional. 55
Figura 27. Diagrama del algoritmo de detección de camino de cultivo mediante agrupación de píxeles.…. 56
Figura 28. Detección de centroides por acumulación de píxeles. 57

Figura 29. Detección de surcos de cultivo por centroides de acumulación de píxeles. 57
Figura 30. Diagrama del proceso de detección de límites de cultivo. 59
Figura 31. Límites (inicio y fin) de las líneas de cultivo. 59
Figura 32. Detección de centro de surcos usando límites de cultivo. 60
Figura 33. Mapeo inverso de perspectiva aplicado a imagen de cultivo. 62
Figura 34. Diagrama del entrenamiento e implementación del algoritmo YOLOv4. 65
Figura 35. Diagrama del uso de algoritmo YOLOv4 en simulación. 65
Figura 36. Etiquetado de clases en imágenes. 66
Figura 37. Data Augmentation aplicado en dataset. 67
Figura 38. Detección realizada por algoritmo YOLOv4 en imágenes de prueba. 70
Figura 39. Diagrama que describe el algoritmo A*. 72
Figura 40. Diagrama del proceso de evasión de obstáculos en el que se implementa A* para corregir el camino trazado para el dron. 74
Figura 41. Ajuste de trayectoria para el dron a partir de evasión de obstáculos. 75
Figura 42. Comunicación de equipo de simulación y equipo con algoritmo YOLOv4 para obtener coordenadas de objetos detectados. 78
Figura 43. Diagrama del proceso de evasión de obstáculos en simulación en Gazebo. 80
Figura 44. Mapa de elementos en cultivo a partir de datos de simulación. 81
Figura 45. Desplazamiento del dron desde condiciones iniciales. 82
Figura 46. primer y segundo desplazamiento del dron. 83
Figura 47. Desplazamiento del dron en el que se presenta una perturbación. 83
Figura 48. Gráfica de error para cada desplazamiento. 84
Figura 49. Velocidades de los motores del dron al desplazarse desde condiciones iniciales. 85
Figura 50. Velocidades de los motores del dron al desplazarse hacia el segundo punto. 85

1. Introducción

Colombia es un país que cuenta con los distintos pisos térmicos existentes, lo que ha permitido que la actividad agricultora se desarrolle a lo largo del tiempo, convirtiéndose en el sustento de millones de personas y en un importante factor de crecimiento de varias regiones, el 17% de la fuerza laboral del país es representada por personas que se dedican específicamente a la agricultura, no solo empleándola para dar alimento en este país sino también para abastecer a otras naciones y generar comercio mediante la exportación de diferentes productos [1]. Es de gran importancia explotar adecuadamente este recurso para que así, los productos puedan ser de calidad, con la posibilidad de reducir impactos ambientales y aumentar la eficiencia de los procesos (es posible reducir la cantidad de químicos y contaminantes y por ende el deterioro del suelo y del cultivo). En la actualidad se hace uso de tecnologías como sensores y maquinaria para monitorear, controlar variables involucradas en el estado de los cultivos y optimizar los diferentes procedimientos que se llevan a cabo en la denominada "agricultura de precisión" [2][3]. Se han realizado investigaciones en las que se procesan datos de cultivo obtenidos mediante diferentes sensores, se emplean algoritmos relacionados a estos para conocer y hacer uso de las propiedades y el estado de cultivos de manera que se toman acciones para hacer más eficiente el uso de recursos dentro de los procesos de agricultura.

El grupo GIDAM desarrolló la plataforma robótica Ceres_AgroBot desde la cual se llevarían a cabo labores de inspección de cultivo y remoción de malezas; para apoyar este proceso se decidió hacer uso de un vehículo aéreo no tripulado desde el que mediante una cámara se dará información de las líneas de cultivo y de obstáculos presentes en este.

Se parte del hecho de que el dron no tiene información acerca de su entorno ni del cultivo. Entonces ¿Cómo podría ubicarse y desplazarse a lo largo del cultivo, incluso con elementos presentes que pueden presentarse como obstáculos?

Diferentes algoritmos para ubicar un vehículo en su espacio han sido investigados, dentro de estas se encuentra el SLAM (*Simultaneous Localization And Mapping*), el cual, mediante datos obtenidos por sensores, hace que las plataformas móviles sean capaces de ubicarse en su espacio y de generar un mapa de su entorno [4]. Para aplicarlo se han utilizado diferentes algoritmos y filtros que aprovechan los sensores disponibles en cada caso de uso, por lo que dependiendo de las necesidades se han usado distintas versiones de SLAM como lo son RGB-D SLAM [2][3], RBPFSLAM [5], FastSLAM [6], entre otros.

Para el caso de este proyecto, se tomó la idea general del SLAM y a partir de la sensórica embarcada en el dron, se identifica el camino que el dron recorrerá a medida que almacena en un mapa información de líneas de cultivo, camino recorrido y objetos presentes en el terreno.

Parte de la literatura investigada para este proyecto fue tomada como inspiración para establecer los diferentes cambios y operaciones que se realizarían a las imágenes del terreno

con el objetivo de detectar las líneas de cultivo. En este caso se realizaron segmentaciones por color, filtros de promedio y operaciones morfológicas que definieron los caminos de cultivo verde presentes en la imagen. Dada la variedad de texturas y elementos presentes en el cultivo, se usó una red convolucional para identificar elementos del terreno como bolsas y plantas que no fueron segmentadas como parte del cultivo.

Teniendo esta segmentación se detectaron los inicios y finales de cada línea de cultivo, para finalmente conservar solamente aquellos datos que se encontraran alineados al camino que se desea recorrer. Con este camino identificado se estableció una trayectoria a partir del cálculo de los puntos medios del inicio y final obtenidos previamente.

Adicional a esto, en este mismo cultivo se encontraron objetos que, dado que solamente se empleó una cámara, fueron tomados como obstáculos en el plano X,Y. Para que estos fueran detectados por parte del dron se implementó el algoritmo de detección de objetos YOLOv4, con el que se obtuvo información de posición y tamaño de los objetos, para finalmente usar el algoritmo de búsqueda A* con el que se ajustó la trayectoria calculada para el dron de forma que estos objetos pudieran ser evadidos.

Dentro del texto se presenta el proceso detallado de cómo se llevó a cabo la implementación de los algoritmos mencionados previamente para que, mediante una simulación en el software Gazebo, se realizara el desplazamiento del dron y la identificación de los elementos clave en el cultivo. En este trabajo se da a conocer el planteamiento del problema (sección 2), el objetivo general y objetivos específicos (sección 3), el estado del arte (sección 4).

Posterior a esto, en la sección 5 se presentan los elementos que fueron requeridos para realizar la simulación del dron y se dan definiciones con respecto a los software empleados (Gazebo y ROS). La sección 6 habla de la etapa de detección de cultivos verdes mediante filtros y procesamiento de imagen. En la sección 7 se dan a conocer las estrategias que se utilizaron para atacar el problema de poseer distintas texturas y elementos que representan al cultivo como bolsas sobre las que crecen las plantas, pero no se encuentran suficientemente pobladas para ser distinguidas por la etapa de filtrado. En la sección 8 se explica el algoritmo ideado para identificar los inicios y fin de cultivo teniendo como base las imágenes previamente procesadas. En esta sección también se indica el proceso realizado para describir la trayectoria que debe recorrer el dron. En la sección 9 se expone el algoritmo YOLO con el que se realizó la detección de los objetos en el cultivo, que es base de la etapa de evasión de objetos descrita en la sección 10. Para concluir el contenido, la sección 11 explica el proceso de integración entre los distintos algoritmos desarrollados con la simulación, teniendo en cuenta que para la ejecución del proyecto se requirió de dos máquinas distintas debido a temas relacionados a las versiones de los lenguajes manejados. En una máquina se realizó la simulación y en otra con mejores características se implementó el proceso de detección de objetos.

En la sección 12 se habla del proceso de mapeo a partir de los datos obtenidos en la simulación, en la sección 13 se presentan los resultados del recorrido del dron durante la simulación, y en la sección 14 se concluye el proyecto, exponiendo recomendaciones para el uso de este en trabajos futuros. Finalmente se presenta la bibliografía en la sección 15.

2. Planteamiento del problema

Para los diferentes equipos que llevan a cabo tareas orientadas a la agricultura de precisión es necesario el conocimiento sobre el espacio en el que se trabajará, si se posee más información del entorno trabajado es posible reducir los posibles errores e inconvenientes al momento de realizar las tareas designadas. Si bien se deben tener algoritmos para reconocer el terreno en un equipo que realiza labores en un cultivo, al trabajar en conjunto con equipos que se enfocan en obtener información sobre este terreno se tienen más posibilidades de detectar elementos del cultivo [7][8], obstáculos e inconvenientes a lo largo del camino que se debe recorrer. Dada la necesidad del sector agrícola por implementar una agricultura de precisión en la que se incluya tecnología para realizar procesos de forma autónoma y eficiente, de manera que se minimice el uso de recursos y el efecto nocivo sobre el suelo de cultivos, y a su vez se maximice la calidad de las cosechas, es necesario preguntarse:

¿Cómo podrían usarse teorías de tratamiento de imágenes para la identificación de surcos en un cultivo, para la planificación de la trayectoria, su seguimiento y posterior mapeo, soportado desde la sensórica embarcada en un dron?

3. Objetivos

En esta sección se describirán los objetivos planteados para el proyecto, sobre los cuales se desarrolló la propuesta y la implementación de cada uno de los elementos expuestos en los posteriores capítulos.

3.1. Objetivo General

Identificar los surcos en un cultivo simulado para generación, seguimiento de trayectoria y posterior mapeo desde un vehículo aéreo no tripulado, usando visión artificial y algoritmos para el procesado de la información obtenida.

3.2. Objetivos Específicos

- Implementar un sistema de visión sobre el vehículo aéreo para identificar surcos y obstáculos en un cultivo simulado.
- Aplicar algoritmos de procesamiento de imagen que permitan usar los datos obtenidos por el sistema de visión para generar una trayectoria y un mapa.
- Implementar un algoritmo para que el dron explore el cultivo mediante el seguimiento de la trayectoria generada.
- Realizar pruebas en simulación para revisar el desempeño de los algoritmos de detección y generación de trayectoria, desde el error de seguimiento de las trayectorias generadas con base a la ubicación de los surcos.

4. Estado del arte

En un contexto globalizado y con una población creciente, la escasez de alimentos puede llegar a ser un problema latente. Por lo anterior, la agricultura de precisión surge como un elemento necesario para asegurar procesos eficientes en la generación de alimentos. Con el creciente uso de sistemas de posicionamiento global GPS, los agricultores han ganado conocimiento inmediato de la posición o zonas de interés en sus cultivos, consiguiendo monitorear el crecimiento del cultivo y realizar el mantenimiento del suelo y demás factores ambientales [9].

El crecimiento productivo de Colombia en materia de hortalizas, la han situado entre los cinco primeros países de América Latina. El grupo GIDAM ha incursionado en proyectos dedicados a la vigilancia con plataformas aéreas [10], tareas de formación de robots [11], incluso, el desarrollo de herramientas para la preparación del suelo, conservación de cultivos [12] y detección de malezas [13].

Dentro de los proyectos desarrollados en el grupo GIDAM, se encuentra la plataforma Ceres_AgroBot (ver Figura 1), la cual quiere ser dotada de autonomía usando la teoría de SLAM, planificación de trayectorias, seguimiento de trayectorias y automatización de procesos, para que asista las labores de agricultura de precisión, mediante el embarque de sensores que le permitan realizar su localización y mapeo simultáneo, detectar y evadir obstáculos durante su desplazamiento, diagnosticar el estado de salud del cultivo, remover maleza y aplicar insecticidas/nutrientes.

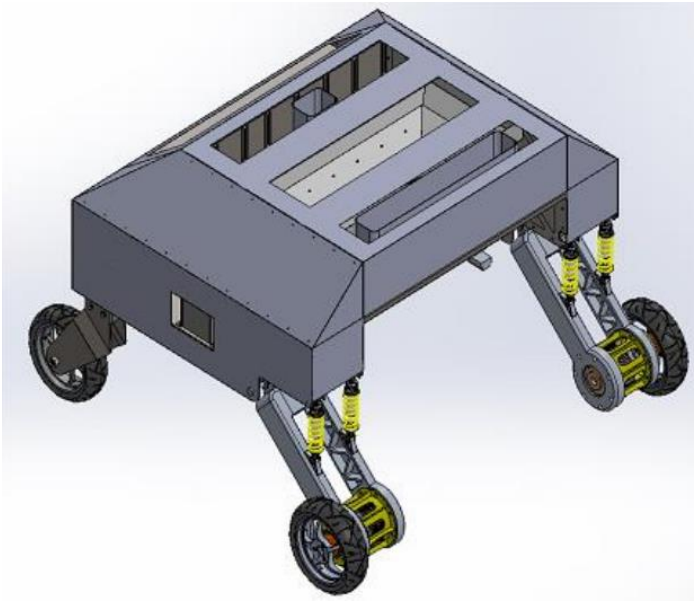


Figura 1. Ceres_Agrobot, Robot dedicado a labores de agricultura de precisión -Desarrollo dentro del grupo GIDAM en proyecto investigativo orientado a plataformas que permitan hacer diagnóstico e intervención de cultivos.

Considerando lo anterior, y teniendo en cuenta que una vista aérea puede dar una perspectiva global del terreno a explorar, se hará uso de un UAV (dron) de referencia Matrice 600 (ver Figura 2), el cual estará provisto de sensores y un sistema embebido que permitirá, por medio de un algoritmo de detección de cultivos que emplea procesamiento de imágenes, sobrevolar el cultivo y recorrerlo para proporcionar información del terreno y de posibles obstáculos con los que se pueda encontrar la plataforma Ceres_Agrobot.



Figura 2. Dron Matrice 600. Hexacóptero diseñada para aplicaciones industriales y fotografía aérea profesional [14].

De esta forma, se le proporcionará al robot terrestre una posible trayectoria para ejecutar las labores de agricultura mencionadas en el párrafo anterior, teniendo en cuenta objetos que puedan obstruir su camino.

4.1. Aplicación de los drones en investigación

El uso de UAVs en la actualidad se ha dado con propósitos de investigación en los que se exploran distintos algoritmos y uso de sensórica para resolver una necesidad. Dependiendo de la aplicación (monitoreo, toma de datos, toma de muestras, entre otras) el tipo de vehículo móvil empleado suele ser aéreo o terrestre. En el caso de los vehículos aéreos como los drones se han logrado avances últimamente para su uso en diferentes industrias, lo cual trae ventajas favorables debido a que se logran superar obstáculos con los que siempre se ha tenido que

lidiar en el caso de vehículos terrestres. El uso de drones se ha evidenciado en el sector militar, la industria agrícola en la cual generalmente se hace uso de pesticidas [15], e incluso en el área de servicios y entretenimiento.

Una gran ventaja de los drones es el manejo del espacio que se les puede dar; para estructuras pequeñas existen drones acordes a este tamaño que son capaces de navegar sin problema y para un ambiente al aire libre existen drones más robustos con capacidad de soportar distintas condiciones climáticas. El uso de estos vehículos en exteriores da una vista diferente en el entorno con la cual se puede trabajar y tomar datos que, por ejemplo, pueden ser usados para distinguir entre diferentes tipos de vegetación [7] o entre técnicas de arado usadas en un cultivo [8].

En Colombia hay variedad de empresas que usan drones para prestar servicios de fertilización, aplicación de fungicidas y toma de fotografías para cartografía, levantamiento de terrenos [16] y análisis de zonas de cultivos [17], sin embargo, esto es realizado con pilotos a los que se requiere capacitar de forma tal que no haya daños en el procedimiento. Las aplicaciones antes mencionadas son parte de la denominada agricultura de precisión que ha crecido bastante en los últimos años, en esta se hacen uso de sensores como cámaras RGB-D, termómetros, redes de sensores y datos satelitales que brindan información muy útil para monitorear la actividad, el estado y las necesidades de las plantas [2][3], optimizar la forma en que se realizan ciertos procesos, entre otras características que favorecen la competitividad y el desarrollo tecnológico para esta industria.

4.2. Algoritmos usados en conjunto con UAVs

Para este proyecto es necesario saber no sólo en qué lugar se encuentran las líneas de cultivo, sino también en qué lugar se encuentra el dron para poder indicar cuánto se debe desplazar para cada trayectoria y a su vez tener en cuenta esta posición en el proceso de evasión de obstáculos.

Esta clase de proyectos con robots en entornos en los cuales se requiere conocer su posición ha llevado al desarrollo de distintos algoritmos que involucran el uso de sensores.

4.3. SLAM

Es el método de localización y mapeo simultáneo (SLAM), el cual es actualmente objeto de investigación en el área de robótica móvil. Por las características que este permite, con datos obtenidos por medio de los sensores para realizar el mapeo correspondiente del lugar donde se encuentra el robot, no solo se mejora la orientación de un operador humano con respecto

a la estructura en la que este se encuentra, sino que también, estos mismos datos son usados para mejorar la estimación de la posición del robot. En ausencia de un mapa, es posible que existan errores o desviaciones con respecto al valor de la posición real, mientras que, con este, es posible eliminar esta clase de errores cuando se regresa a puntos clave o puntos de referencia establecidos durante el mapeo (proceso denominado lazo cerrado) [4].

El SLAM es usado tanto en entornos cerrados, o en aquellos en donde no se dispone de sensores o sistemas de ubicación avanzados, como en lugares abiertos donde es posible usarlos para mejorar la capacidad de este algoritmo. En distintas universidades de Colombia se han realizado investigaciones referentes a SLAM en las que se hace integración de sensores para aplicarlo a robots móviles de tipo terrestre, evaluando este algoritmo con distintos tipos de filtros [18], se han implementado sistemas de navegación autónomos para reconstrucción de terreno [19], también se ha usado con propósitos de mapeo en entornos semi-estructurados donde se usan líneas rectas como puntos de referencia [20].

El mapeo puede ser realizado en 2D y consiste en representar áreas y lugares que el robot en cuestión ya haya recorrido. Investigaciones de este tipo ya han sido llevadas a cabo en las que dadas las limitaciones de sensores como el de ultrasonido, se hace uso de kinect para la creación del mapa en dos dimensiones simultáneamente y hacer uso de este mapa para regresar a una posición de “home” [21], el mapeo por parte de SLAM también puede ser combinado con otros algoritmos para obtener mapas en 2D con información del terreno, por ejemplo la detección de árboles y frutas utilizando un Kinect y la obtención de un mapa generado por SLAM haciendo uso de paquetes de ROS para posteriormente integrar estos datos y tener un mapa de una área agrícola con información sobre los árboles que se encuentran allí [22]. Similar a esto, han sido usadas cámaras de video monoculares y nubes de puntos en 3D para el mapeo de cultivos de piña incluyendo reconstrucciones en 3D de las frutas y de sus ubicaciones [23].

Actualmente el SLAM es empleado para labores de investigación o de reconocimiento de estructuras y es posible combinarlo con otros algoritmos para cumplir con ciertos objetivos, pero no son muchos los casos en que se haga uso de este algoritmo para industrias como la agrícola, en algunos casos hay mapeo del terreno, se identifica la posición mediante un GPS o incluso una trayectoria de vuelo autónomo predispuesta [24], pero no hay un vuelo autónomo como si se realiza en SLAM.

Dentro de la solución al problema de mapeo y localización, el tipo de algoritmo empleado dependerá de la complejidad del problema a resolver, debe evaluarse el consumo de tiempo, de recursos, la codificación, los sensores disponibles, la dinámica a la que se verá sometido el robot y el ambiente de trabajo. Dependiendo de la situación y considerando los elementos disponibles para atacar una necesidad en específico se han desarrollado diferentes versiones u optimizaciones a SLAM (RGB-D SLAM, RBPFSLAM, FastSLAM, RO-SLAM, EIF SLAM, EKF SLAM), cada una implementa sus respectivos algoritmos los cuales trabajan de forma más efectiva según la aplicación que se le dé. Cada una de estas versiones tiene exigencias respecto al hardware a utilizar y al consumo computacional que se empleará, por lo cual, al momento de seleccionar la técnica más adecuada se debe tener en cuenta que tan

necesarias serán las características que estos algoritmos ofrecen con el fin de cumplir con el propósito del mapeo y localización sin afectar el rendimiento y la correcta ejecución de la operación. La teoría de filtrado es parte importante de SLAM, se han dado diversas aproximaciones para mejorar la estimación de los valores de posición y del mapeo. El Filtro de Información Extendida (EIF) es usado en el EIF SLAM, El filtro extendido de Kalman (EKF) es usado en el EKF SLAM mientras que el filtro de partículas es usado en el RBPF-SLAM y en FastSLAM, lo cual lo hace más flexible al permitir diferentes distribuciones de ruido además de la Gaussiana [6]. Para el caso del algoritmo RBPF-SLAM, se usan muestras de partículas para representar la distribución de probabilidad de los resultados de localización, de esta forma cada partícula representa la posible posición en la que se encuentra el robot [5].

Cuando se trabaja con FastSLAM 2.0 las estimaciones de posiciones y los mapas son separados. La esencia de este método es que se trabaja con el filtro de partículas de Rao-Blackwellized para estimar la posición y se usa EKF para estimar los mapas. A partir de la implementación de filtros que permiten optimizar el SLAM, es posible mejorarlo aún más haciendo uso de algoritmos y de métodos. El QPSO (*Quantumbehaved Particles Swarm Optimization*) lleva el comportamiento cuántico al algoritmo PSO, que es usado para encontrar rápidamente soluciones exactas o aproximadas a problemas de optimización; al juntar el FastSLAM con el QPSO se busca optimizar la posición estimada de partículas, que hace que las posiciones estimadas sean más cercanas a las verdaderas [4]. Los sensores son parte esencial del proceso de localización, los resultados de las pruebas y prácticas varían según su calidad, calibración y buen funcionamiento, diferentes técnicas se han realizado para sacarle provecho a estos implementos, como el uso de redes de sensores, que por medio de nodos (dispositivos autónomos) recogen muestras de un ambiente para ser transmitidos a una locación [6].

El sensor Kinect se ha usado en diferentes aplicaciones con el que es posible establecer puntos de referencia visuales para determinar el camino de un robot y la construcción de un mapa [25]. Sin embargo, el uso de un solo sensor para una tarea involucra limitarse a sus únicas características y por ende, a sus desventajas. Por ejemplo, cuando se trabaja con instrumentos visuales es posible que los movimientos, la dinámica, ambientes similares o *aliasing* de percepción incidan y afecten negativamente los resultados. Integrar otro sensor al sistema (como lo puede ser una IMU) permitiría predecir y solucionar problemas relacionados con la dinámica del sistema que producen errores en la percepción, haciendo posible el estudio en escenarios que usando el SLAM por sí solo no podría ser realizado [26].

Las cámaras como instrumentos en el SLAM basado en visión son usadas debido a sus pocas limitaciones con respecto al rango, dirección y peso en comparación con sensores de ultrasonido y de infrarrojo, sobre todo para aplicaciones en que los robots móviles son pequeños [27]. El uso de cámaras enfocado a SLAM es una situación usual; la implementación de RGB-D SLAM involucra cámaras RGB-D con las que se pueden

construir mapas 3D de nubes de puntos en tiempo real [28]. A pesar de que el uso de las cámaras y dispositivos visuales trae grandes ventajas al momento de aplicar SLAM, estos dispositivos no son esenciales, otra forma de aplicar este procedimientos es mediante *radio beacons*, los cuales se distribuyen en el escenario con el objetivo de construir un mapa de un lugar desconocido y para la autolocalización en este mapa, a este método se le denomina *Range Only Simultaneous an Localization Mapping* (RO-SLAM) y trae distintas ventajas en ciertos casos dado que no depende de las condiciones de luz y resuelve varios problemas del SLAM visual.

Dadas sus características, SLAM permite ser usado en una amplia cantidad de aplicaciones, y para cada una de estas en las que se dispone de dinámicas, estructuras, recursos (sensores, actuadores) y ambientes específicos existen algoritmos que permiten una solución óptima del problema a tratar. Además de esto, aún es posible innovar en el campo de la robótica móvil enfocado a este método debido a la gama de sensores y procedimientos disponibles, de esta forma se facilitarían las tareas para la industria y se permitiría a las tecnologías actuales usar su capacidad lo más posible.

4.4. Procesamiento de imagen

El procesamiento de imagen enfocado en los cultivos ha sido utilizado en su mayoría para diferenciar entre especies de cultivos, distinguir entre las plantas y el fondo (tierra, elementos ajenos al cultivo) y determinar el estado de los cultivos. Esta práctica se vuelve indispensable para el proyecto planteado dado que en el cultivo de la universidad existen distintos elementos que deben ser identificados tanto para establecer cuál es la línea de cultivo que el robot va a seguir y donde se encuentra (plantas, tierra de los surcos) como para evadirlos modificando la trayectoria (obstáculos).

El uso de técnicas relacionadas al color de las imágenes es más efectivo para cultivos con colores distinguibles y únicos [29]. Los filtros de color han sido ampliamente usados para segmentar elementos en el cultivo cuyo color es característico y permite ser diferenciado del resto, incluso para casos en los que los cultivos se encuentran en ambientes al aire libre expuestos a la luz del sol donde la intensidad lumínica varía. En situaciones así se requiere hacer uso de más de un filtro. En caso de tener un cultivo con diferentes plantas, en las que sus características difieren entre ellas, el procedimiento a realizar para la detección de cultivos tampoco se puede limitar solamente a un filtro de color o a una sola técnica que emplee esta característica.

Desde hace más de una década se realizan estudios relacionados al uso del color para binarizar imágenes y a partir de esto, realizar transformaciones y cálculos que permitan efectuar un análisis de los cultivos que se manejan. En [30] se tomaron fotografías de diferentes plantas y se llevó a cabo la conversión de imágenes en color a escala de grises a partir de índices de contraste, para posteriormente hacer un análisis, con el que se identificaron como plantas aquellos objetos que superaban un límite de tamaño de píxeles y un límite de valor de grises. A partir de estos análisis se calcularon datos a cada planta: redondez, aspecto, perímetro, alargamiento y primer momento central invariante. Estos datos permitieron diferenciar entre plantas monocotiledóneas y plantas dicotiledóneas y a su vez, describir la forma de la planta.

Durante las investigaciones en las que se hace uso del color para procesar imágenes se han comparado diferentes técnicas existentes y también se ha tratado de realizarles mejoras. Por ejemplo, en [31] se explica la creación de un índice de vegetación denominado *Excess Green minus Excess Red* (ExG-ExR) que tiene incluido un umbral fijo, por lo que no requiere hacer uso de un método extra para binarizar la imagen. Este índice es comparado contra los métodos de *Excess Green* (ExG) y los índices de diferencia normalizados (NDI) que son combinados con el método de Otsu. Las pruebas fueron realizadas con imágenes de plantas de diferentes especies y en diferentes etapas de crecimiento: amarantos, soya y terciopelo. Con los resultados obtenidos se determinó que el método (ExG-ExR) posee un mejor índice de calidad.

También se han implementado proyectos que hacen uso del espacio de color para detectar características del cultivo y la maleza, en los que la segmentación de imágenes es indispensable. Para el caso de [32] en donde al analizar los espectros de color verde relacionados a las plantas de maíz, se observa que estos son contaminados por otros espectros de color relacionados a materiales de la tierra luego de las lluvias, lo que generaría que se identificara incorrectamente las plantas si se usasen métodos solamente relacionados con la identificación del color verde. Por esta razón durante este proyecto se implementaron máquinas de vectores de soporte (SVM) que permiten identificar plantas con espectro de color verde cubiertos por otros elementos y descubiertos. El proceso consistió en:

1. La identificación del verde y posterior umbralización mediante el método de Otsu para obtener una imagen binaria con dos clases.
2. La identificación de los vectores de soporte para cada clase.
3. El cálculo de valores promedio para cada conjunto de vectores de soporte con los que se establecen los límites de las plantas cubiertas y no cubiertas por la tierra.

Este proceso fue lo suficientemente robusto para soportar condiciones de diferente iluminación y permite identificar plantas que han sido contaminadas con materiales del suelo, debido a irrigación artificial o a lluvia natural. Este proyecto es importante y bastante relacionado con el uso del color, porque identificando la pérdida de color verde, también es posible monitorizar el cultivo luego de algún tratamiento, tanto efectividad de tratamientos contra las malezas, como afectación al cultivo por parte de estos.

A partir de los colores y de las características de la imagen, es posible no solo segmentar lo que pertenece a plantas o al suelo, sino también identificar texturas que se relacionan a cada clase de imagen. En [33] combinan diferentes estrategias para dividir en tres clases principales las imágenes: plantas, cielo, y suelo. Esto se hace mediante métodos de umbralización y clusterización difusa que permite identificar texturas automáticamente. Los métodos de umbralización se realizaron tomando como valor umbral el valor medio de las imágenes de intensidad gris, para separar las texturas principales se requirió hacer uso del color verde para identificar las plantas, con los métodos ExG, CIVE, ExGR, y VEG. Mientras que para la identificación del suelo se usó el color rojo con el método ExR. Para la detección del azul del cielo se usó el método denominado ExB. Posteriormente, para mejorar la clasificación de estas clases, se implementó un clasificador difuso que permitiera catalogar diferentes tipos de plantas basadas en los tipos de verde obtenidos, y reconocer aquellas plantas que erróneamente fueron clasificadas como suelo debido al proceso de adquisición de las imágenes. Este método funcionó correctamente bajo diferentes condiciones de iluminación.

En casos en donde se requiere trabajar imágenes en tiempo real, es necesario llevar a cabo estrategias y métodos que realicen los análisis de imágenes lo más rápido posible sin afectar en gran medida los resultados que se desean obtener. Por ejemplo, en [34] se planteó como objetivo diferenciar entre cultivos y maleza con video en tiempo real, proceso que se podría usar en aplicaciones relacionadas a tratamiento de cultivos en donde se requiera examinar continuamente las características de estos. El proceso de toma de imágenes se realizó a 25 fps y se dividió en los siguientes pasos:

1. Segmentación para discriminar los píxeles de vegetación usando una combinación lineal de planos RGB cuyos coeficientes fueron calculados usando una optimización de algoritmos genéticos.
2. Umbralización mediante intensidad media de píxeles que, según pruebas realizadas en este artículo, resulta ser más preciso y rápido que el algoritmo de Otsu.
3. RCRD (*Robust Crop Row Detection*). Un algoritmo que no depende del tiempo, en el que se acumulan varios fotogramas y mediante operaciones AND entre estos fotogramas, y uso de operaciones morfológicas se filtran las líneas de cultivo.
4. FIP *(Fast Image Processing)*. Con este algoritmo y junto con los resultados de RCRD se identifican las malezas. FIP toma las imágenes segmentadas y calcula de un vector que depende de divisiones en “tiras” que se le realizan a la imagen, y el color de los píxeles en estas divisiones. A partir de este vector y de un valor umbral, se identifican los píxeles relacionados con la vegetación. Usando estos cálculos se obtiene una identificación de la vegetación con FIP, que al validarla junto a las líneas de cultivo en la imagen obtenida por RCRD se establece que aquellos píxeles que corresponden a la vegetación, pero no corresponden al cultivo serán tomados como maleza.

Durante este proyecto fue clave la segmentación de la vegetación para enfrentar la diferente iluminación y las diferencias entre los videos tomados, dado que se hizo uso de seis videos de distintos cultivos en distintos tiempos.

Diferentes técnicas de procesamiento de imagen se han manejado para la detección de cultivos, además de los filtros ya existentes o del manejo de texturas, también es posible realizar operaciones morfológicas en la imagen. En [29] realizan operaciones de erosión y dilatación de imagen para segmentar cultivos de coliflor bajo diferentes condiciones de luz, haciendo uso del espacio de color HSV con el que se buscó diferenciar entre cultivo, maleza y suelo. El proceso se llevó a cabo de la siguiente manera:

1. Conversión de color de imagen de RGB a HSV
2. Remover los píxeles de fondo al comparar los valores de matiz (componente H) con los píxeles de las plantas
3. Se eliminaron los píxeles de trigo que se superpusieran con hojas verdes haciendo uso de los valores de HSV.
4. Binarización mediante valores umbral
5. Eliminación de objetos con área menor a 100 píxeles mediante operaciones morfológicas para aislar las plantas verdes.
6. Encontrar los contornos de las plantas mediante los píxeles continuos en el perímetro de la imagen resultante del paso 5.
7. Calcular el área y centroide del contorno mediante el método de los momentos.
8. Calcular el perímetro del contorno, a partir de este valor y el del área se establece que fue detectada una planta y se dibuja un rectángulo rojo alrededor de esta.

Este método funcionó correctamente bajo diferentes condiciones de luz al validar un rango adecuado de HSV en las que solamente el matiz (componente H) varía dependiendo de la iluminación.

Esta clase de investigaciones en las que es posible identificar la posición de una planta, favorecen el desarrollo de la automatización dentro de la agricultura, al permitir obtener coordenadas que pueden ser usadas como guía para los elementos que realizan las fumigaciones.

4.4.1. Incidencia de las condiciones de iluminación

Como ya se ha presentado, en distintos proyectos se ha buscado dar solución a la detección de los cultivos en más de una condición de iluminación, dado que no basta con realizar una segmentación por color si estas condiciones van a ser variadas, que por lo general sucede cuando el cultivo se encuentra en el exterior con luz natural, donde la nubosidad hace que la luz varíe. Ejemplos de investigaciones en las que se ha tenido en cuenta estas variaciones se encuentran a continuación:

Environmentally Adaptive Segmentation Algorithm (EASA), diseñado para aprender de las imágenes tomadas en diferentes campos y a distintas horas del día, esto incluye las condiciones de color y la variedad de plantas que puede haber en el cultivo. En este caso se realizó un proceso de *clustering*: *partial-supervised-learning* (PSL), que toma un conjunto de semillas seleccionado, para que al momento de realizar el proceso de *clustering*, los píxeles sean agrupados alrededor de estas semillas hasta que finalmente se llegue a un número de clases determinado. Posteriormente se verifica la precisión de la clasificación y se decide si se deben ajustar las semillas o continuar con un proceso en el que se crea un clasificador Bayesiano para posteriormente generar una "Look-Up-Table" para la segmentación [35].

Mean-shift algorithm with back propagation neuronal network (MS-BPNN), en el cual se toman características de cada píxel en la imagen: componentes Matiz y Saturación del espacio de color HSV, los componentes Red y Green del espacio de color RGB y la posición en X y Y. A partir de estos datos se aplica una segmentación de imagen Mean-Shift, para posteriormente hacer uso de una red neuronal que identifica cuál grupo de píxeles pertenece a un cultivo y cuál no. Esta red fue entrenada a partir de datos de áreas de vegetación verdes y no verdes [36].

Mean-shift algorithm with Fisher Linear Discriminant (MS-FLD), este nuevo método toma una combinación del color y coordenadas espaciales de los píxeles para el proceso de segmentación, tomando el espacio de color LUV. El algoritmo combina la capacidad del método de discriminante lineal de Fisher para trabajar con problemas de clasificación de dos clases, con el algoritmo Mean-shift (MS). MS se usa para pre-segmentar la imagen en diferentes regiones, la media del color de cada región se procesa y es proyectada en una dirección "d", para posteriormente compararla con un offset "b", dependiendo de su relación con este offset se marcan los píxeles de esa región con un '1' (blanco) para la vegetación o con un 'o' (negro) para el fondo. Como resultado, MS-FLD tiene un mejor y más estable rendimiento en momentos en los que la vegetación tiene o no tiene sombras [37].

Decision Tree based Segmentation Model (DTSM), este algoritmo parte de la necesidad de analizar plantas que dependiendo de las condiciones de luz pueden verse más ensombrecidas o verse más iluminadas. A partir del algoritmo CART se crea un árbol de decisión para el proceso de entrenamiento y se compara el rendimiento con métodos como ExG, ExG-ExR.

El proceso se realizó con los siguientes pasos:

1. Se obtuvieron imágenes para entrenar el modelo posicionando una cámara con vista sobre una planta de trigo, se tomaron imágenes con periodos de 1 hora, durante diferentes días. A partir de estas imágenes se seleccionaron regiones de interés (ROI) relacionadas a la planta y al fondo, cada una con diferentes condiciones de luz. Cada píxel tiene información de diferentes espacios de color (RGB, YCbCr, HSL, HSV, CIEL*a*b* y CIEL*u*v*).

2. La técnica de selección de características "*wrapper*" fue usada para identificar cuáles de las distintas componentes de los espacios de color seleccionados realmente contribuyeron al sistema diseñado.
3. Se entrenó el modelo para crear un árbol de decisiones con el *dataset* obtenido.
4. La segmentación fue evaluada con el árbol de decisiones sobre imágenes de prueba.
5. Las imágenes segmentadas fueron procesadas para reducir el ruido.

Luego del proceso de segmentación, este resultado se evaluó utilizando imágenes binarizadas manualmente mediante Adobe Photoshop Elements 9. El método evidenció una precisión mayor a otros para imágenes bastante sombreadas o con partes reflectantes. Algo importante a mencionar es que este método puede ser utilizado en otras imágenes sin necesidad de modificar algún valor umbral. Sin embargo, depende bastante de los datos de entrenamiento [38].

Particle Swarm Optimisation clustering and Morphology Modelling (PSO-MM) Este método fue probado con imágenes de cultivos de arroz y de algodón. Consta de una etapa offline y otra online. En la etapa offline se determina el número óptimo de grupos de clasificación, dado que dividir los elementos en solamente dos clases, vegetación y no vegetación, posiblemente no cubriría con precisión los casos en que las plantas se encuentran bajo diferente iluminación. Por otra parte, si hay más grupos de los suficientes esto implicaría un costo computacional mayor. Luego de esto se hace uso de operaciones morfológicas de erosión y dilatación para establecer un modelo de color de la vegetación.

En la etapa online se hace uso de la clusterización K-means basado en el método *Particle Swarm Optimization* (PSO), con el que se buscó aprovechar la alta velocidad de cálculo de K-means y escoger adecuadamente las semillas iniciales de este algoritmo haciendo uso de PSO. Con este algoritmo se agrupan las imágenes entre clases de vegetación y clases de no vegetación. Posteriormente el modelo de color se usa para distinguir las clases de vegetación y dar un resultado de la segmentación.

Los resultados de la aplicación de este método se compararon contra otros métodos: ExG&Otsu, ExGExR, EASA, GMM y ColourHist. Se evaluó la media de calidad de segmentación y la desviación estándar, medidas en las que el método propuesto fue superior a los otros, aunque tiene la desventaja de no ser muy apto para uso en tiempo real debido a los diferentes pasos de procesamiento [39].

Para algunas de las investigaciones que se han mencionado respecto al procesamiento de imágenes de cultivos se usan procedimientos computacionalmente costosos y puede no ser factible aplicarlos en tiempo real. Esto es importante y se tuvo en cuenta al momento de estudiar cuales de estos métodos de detección de cultivos y de segmentación podrían ser implementados en el proyecto.

4.5. Detección de objetos

Además de la detección de los cultivos para generar las rutas del recorrido del dron también es necesario detectar los obstáculos que interfieran con estas rutas para poder evadirlo.

Se han realizado distintas investigaciones relacionadas con detección de objetos, implementando algoritmos que manejan procesamiento de imágenes y redes neuronales. El desarrollo de las redes neuronales convolucionales y el aprendizaje profundo han permitido crear sistemas de reconocimiento de objetos y obstáculos no deseados en ambientes en que la irrupción de estos representa un riesgo.

Foreign Objects Detection Network for Power Substation (FODN4PS): Un algoritmo de detección de objetos utilizado específicamente para reconocer objetos intrusos que pueden generar fallas y paralizar subestaciones eléctricas. Esta es una tarea que actualmente se realiza manualmente y depende de los sensores que se usen, las personas que la inspeccionan y en qué condiciones se realiza la revisión. Consta de dos redes neuronales:

- MORE Net, la cual obtiene la posición de los objetos. El algoritmo analiza los 3 primeros fotogramas del video de entrada, y a partir de este genera un modelo del fondo. De esta forma cuando ingresa un objeto extraño hay un cambio en la imagen, lo que se compara con el modelo de fondo se puede obtener el área del objeto. Esta red es ajustada para reducir el tiempo de cómputo cambiando la manera en la que otras generan las "casilla objetivo".
- Una red de clasificación, que reconoce a cuál categoría pertenece el objeto detectado. Su entrada son las casillas objetivo, generadas por la red MORE y está compuesta por redes convolucionales y una red *Fully Connected Network* (FCN).

Como resultado la red presentada es más rápida y precisa que otras redes contra las que se les comparó: Fast R-CNN y Mask R-CNN. Dado que el desempeño del modelo se ve afectado por la cantidad de imágenes con las que se entrena, dentro de la investigación también se muestra el resultado de diferentes métodos empleados para las imágenes de entrenamiento [40].

Así como se han realizado investigaciones en las que se mejoran los métodos de procesamiento de imagen, en estos casos hay proyectos que han buscado optimizar los modelos que existen actualmente. Un ejemplo es [41] en donde se propone un modelo denominado ***Improved Faster R-CNN*** con el objetivo de detectar múltiples objetos. Para este modelo se hace uso de parte de la arquitectura de la red VGG16 para mejorar la red Faster R-CNN, que no es tan eficiente en la detección de objetos pequeños. Se emplearon *dataset*s como Cityscapes y KITTI, que contienen imágenes del tráfico en ciudades. Para estos *dataset*s el modelo propuesto presentó mejor precisión que los modelos Fast R-CNN y Faster R-CNN, pero en los casos complejos como movimiento de objetos y condiciones climáticas no supera la precisión promedio.

UFOD: Consiste en un *framework* que entrena diferentes algoritmos de detección de objetos, los compara y escoge el mejor modelo. Para su creación se requirió realizar una investigación de los frameworks de detección de objetos que usan aprendizaje profundo, sus características y los algoritmos que ofrecen, dado que no hay ninguno que contenga todos los algoritmos existentes para esta clase de problemas. Para su uso se establece el siguiente flujo de trabajo:

1. Selección de *dataset* de imágenes y parámetros de configuración.
2. División del *dataset* entre set de entreno y set de prueba.
3. Entrenamiento de modelos
4. Selección de modelo basado en el rendimiento y la velocidad.
5. Salida del framework, que puede ser el mejor modelo o un conjunto de modelos. Se entrega una aplicación en forma de Jupyter Notebook para hacer uso del modelo.

Algunos de los algoritmos que soporta el *framework* son Faster-RCNN, SSD, YOLO y Retinanet [42].

YOLOv4 (You Only Look Once): Es un sistema de detección de objetos en tiempo real. El algoritmo consiste en dividir la imagen recibida en una cantidad SxS de celdas, para cada una de estas crea "cajas delimitadoras" a las que se les calcula un nivel de certidumbre. Posteriormente se eliminan las cajas que están por debajo de un valor umbral y luego se eliminan las detecciones duplicadas de un objeto asegurando la más precisa por cada objeto. Esta es la cuarta versión del algoritmo YOLO, la cual tiene un incremento del 10% en precisión promedio y 12% en fps con respecto a la versión previa. Para esta versión se tuvo como objetivo aumentar la velocidad de la red neuronal y optimizarla para la computación paralela, y se implementaron nuevas características como el uso de *Weight-Residual-Connections* (WRC), *Cross-Stage-Partial-connections* (CSP), *Self-Adversarial-Training* (SAT), entre otras [43][44].

Aunque el algoritmo YOLO es usado ampliamente en la detección de objetos en diferentes escenarios, también se ha usado para identificar características de los cultivos. En [45] este modelo se utiliza para detectar enfermedades en las plantas procesando imágenes de hojas a 45 fps en tiempo real. El método empleado fue de la siguiente manera:

1. Obtener un *dataset* de 25 clases entre las que se pueden clasificar imágenes que representan diferentes estados para hojas de manzanas, tomates, maíz, uvas, entre otros.
2. Dada la limitada cantidad de imágenes encontradas para entrenar el algoritmo, se recurrió a hacer un aumento de los datos mediante métodos artificiales como agregar ruido y rotaciones de imagen. También se emplearon redes GAN (redes Neuronales Generativas Adversarias) para generar imágenes nuevas a partir de las ya existentes.
3. Uso del modelo, en el que se establecen dos tipos de casillas delimitadas o *bounding boxes*, una de estas se utiliza para detectar la hoja de la planta, y la otra de estas se encarga de detectar las áreas infectadas.

De esta forma a pesar de las pocas imágenes encontradas de enfermedades de plantas, fue posible hacer uso de técnicas para entrenar adecuadamente un algoritmo que usualmente se emplea para detectar objetos, con el objetivo de identificar esas enfermedades y apoyar a las labores de agricultura haciendo de este un proceso más sencillo para las personas encargadas de los cultivos.

5. De la simulación y sus elementos

Dadas las condiciones de pandemia presentadas durante el periodo en el que se realizó el proyecto, este no se implementó de manera física como se ideó en un comienzo, se le dió enfoque a una simulación completa del proyecto desde la cual se desarrollaron y ejecutaron todos los algoritmos y procesos orientados a la identificación del cultivo y desplazamiento de la aeronave.

Para el proyecto se requirió de un ambiente de simulación en donde fuera posible tener en cuenta las mecánicas y dinámicas tanto del dron como del espacio donde este despegue. Además de simular el movimiento del dron, se toman elementos del simulador como cámaras para llevar a cabo el procesamiento de imagen y se tiene en cuenta la simulación del espacio de cultivo donde se trabajará, la cual será explicada más adelante.

Todo el proceso de simulación incluye los pasos de:

- Preparación del ambiente
- Procesamiento de imagen
- Generación y corrección de trayectoria
- Desplazamiento del dron

5.1. ROS (Robot Operating System)

Es un meta-Sistema Operativo *open source* que provee herramientas y librerías para el desarrollo de software orientado a robótica. En la actualidad es utilizado por estudiantes, investigadores y empresas, lo que hace que una gran lista de robots haga uso de ROS.

Para desarrollar este proyecto se utilizaron los siguientes elementos básicos de ROS que son empleados para la ejecución de procesos y la comunicación entre estos:

- Nodos: Representan el código ejecutable, cada nodo se encarga de controlar un aspecto del sistema y de ejecutar una acción. Pueden publicar información o suscribirse a un *topic*. Estos pueden interactuar entre ellos y ser distribuidos en un solo equipo, varios equipos o entre equipos y robots [46].
- Mensajes: Es una estructura de datos tipada publicada desde los nodos hacia los *topics* [47].
- Topics: son los buses a través de los cuales los nodos comparten la información (mensajes). Un nodo puede publicar información hacia un *topic*, y a su vez, otro nodo puede estar suscrito a ese *topic* para utilizar la información en algún proceso [48].

5.2. El ambiente de simulación

El simulador escogido para esta tarea fue Gazebo, dado que cuenta con numerosos motores de físicas, soporte para *plugins*, variedad de sensores y de cámaras. Los múltiples proyectos que se han realizado con este simulador y la documentación que tiene, favorecieron el desarrollo de este proyecto. También se tuvo en cuenta que de la plataforma Ceres_Agrobot hace uso de este simulador, por lo cual, de ser requerido, será más sencillo migrar datos, algoritmos y elementos de simulación.

5.3. El cultivo en Gazebo:

Para realizar la simulación se requirió crear un nuevo "mundo" en Gazebo, en los archivos del simulador se agregó la imagen completa del cultivo como una "imagen del plano de tierra" o *ground plane image* dentro de la librería de texturas. Para este proceso fue necesario crear un modelo, el cual está conformado por lo archivos:

- *config* en el que se establecieron las configuraciones del modelo.
- *sdf* desde el cual se le dieron las características geométricas y espaciales.
- *material* donde se especificó la textura correspondiente a la imagen del cultivo.

Luego de realizar este proceso, en el nuevo mundo creado denominado "crop2.world" se importó el modelo creado previamente, que se puede observar en la Figura 3.

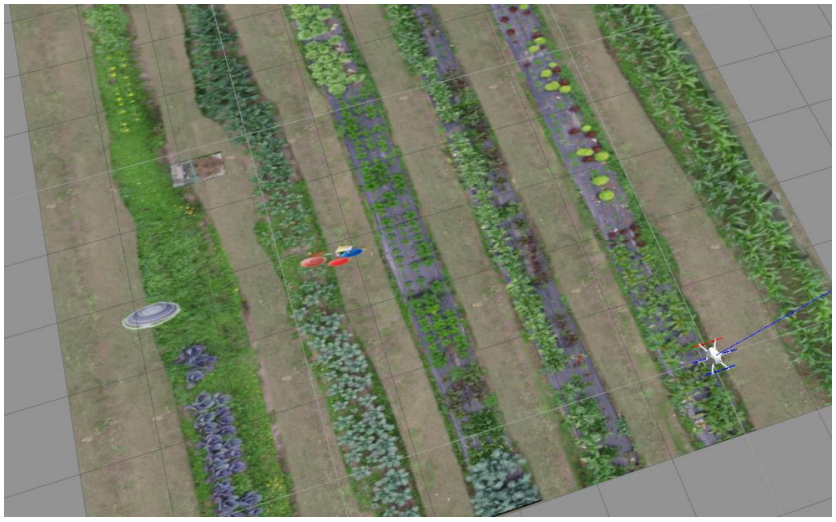


Figura 3. Ambiente de simulación del cultivo en Gazebo.

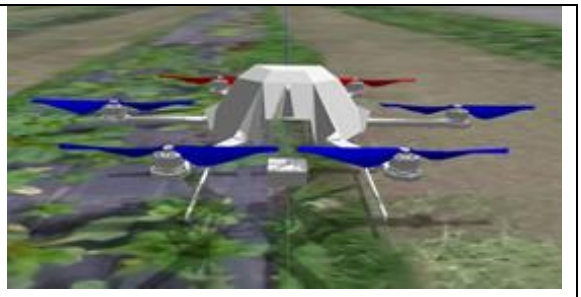
5.4. El hexacóptero

En Gazebo y en la web se encuentran diferentes modelos de drones para ser utilizados en este simulador. Sin embargo, la mayoría de los proyectos que incluyen UAVs hacen uso de modelos de *quadrotor*, que para este caso no es el deseado debido a que se trata de encontrar un dron similar al Matrice 600.

El modelo de hexacóptero que se encontró más similar al Matrice 600 fue el AscTec Firefly, un hexacóptero a una escala más reducida que está incluido dentro del simulador de vehículos aéreos "RotorS". En la figura 4 se presenta el dron físicamente y en el ambiente de simulación.



(a) Imagen del hexacóptero real.



(b) AscTec Firefly en ambiente de simulación Gazebo, proporcionado por RotorS.

Figura 4. Hexacóptero Asctec Firefly usado en el proyecto.

Dentro de Gazebo vienen también sensores que pueden ser agregados a los vehículos, como IMU, sensores de odometría o cámaras.

5.5. Integración con la cámara

El modelo de dron trabajado inicialmente no contaba con la cámara requerida para realizar el proceso de reconocimiento del cultivo. Por esta razón se requirió agregarlo dentro de los archivos en formato *xacro*, los cuales describen los elementos que conforman el dron. Se establecieron los parámetros respectivos:

```
<xacro:camera_macro
  <!—Parámetros relacionados al simulador-->
  namespace="${namespace}"
  parent_link="${namespace}/base_link"
  camera_suffix="nadir"

  <!—características de la cámara-->
  fotograma_rate="60.0"
  horizontal_fov="2"
  image_width="1280"
  image_height="720"
  image_format="R8G8B8"
  min_distance="0.02"
  max_distance="50"
  noise_mean="0.0"
  noise_stddev="0.007"
  enable_visual="true">

    <!—Ubicación de la cámara con respecto al dron-->
    <box size="0.05 0.05 0.02" />
    <origin xyz="0 0 ${-arm_length/4}" rpy="0 ${pi/2} 0" />
```

Se utilizó una cámara simulada de 1280x720 a 60 fps y con formato de imagen RGB ubicada en la parte inferior, en el centro del dron, en la Figura 5 la simulación del dron ya tiene integrada la cámara, al realizar esta configuración el simulador publica la imagen de esta cámara a un *topic.*

5.6. El desplazamiento de la aeronave

RotorS cuenta con ejemplos de cómo encender y hacer despegar los drones. Basta con usar un comando *roslaunch* para ejecutar los ejemplos, también debe ser especificada la aeronave y el “mundo” donde se realizará la simulación. Por ejemplo:

```
$ roslaunch rotors_gazebo mav_hovering_example.launch mav_name:=firefly
world_name:=basic
```

Con este comando se ejecuta el ejemplo en donde se eleva el dron Firefly hasta una altura determinada.

A pesar de los ejemplos existentes, indicarle al dron que se mueva en cierta dirección no es algo tan básico, En [49] se da otro ejemplo de un archivo de tipo launch con el que se le dan posiciones específicas al dron a las que este se dirige con precisión dados los sensores de odometría que simula.

Se realizó un entendimiento de este archivo y se logró replicar la forma en que este hace mover el dron, sin embargo, posteriormente se requirió ajustarlo y reescribirlo como código Python para poder trabajarlo más fácilmente con el resto de los módulos y de códigos usados durante el proyecto, dado que el resto del proyecto fue escrito en Python.

Luego de realizar cada una de las configuraciones relacionadas a la simulación, se presenta en la Figura 5 el gráfico de los nodos *publisher* y *subscriber*, y aquellos *topics* a los que se encuentran relacionados. Al momento de iniciar la simulación, los nodos "gazebo", "gazebo_gui", "firefly/lee_position_controller_node", "firefly_joint_state_publisher" y "firefly/robot_state_publisher" comienzan a correr, estos se encargan de iniciar todo el ambiente, incluyendo el mundo donde está el cultivo el dron y sus características físicas, elevándolo hasta una posición inicial.

Posterior a esto, se inicia el programa para recorrer el cultivo, este programa inicia el nodo "run_crop_detection", este nodo se encuentra suscrito a los topics "firefly/camera_nadir/image_raw" y "firefly/odometry_sensor1/odometry" para obtener la imagen de la cámara y las posiciones globales del dron, respectivamente.

En este programa se realiza toda la etapa de procesamiento de imagen, detección de cultivos, comunicación con otras máquinas, detección de objetos y desplazamiento del dron. Cada subprograma y utilidad dividida en sus respectivos módulos.

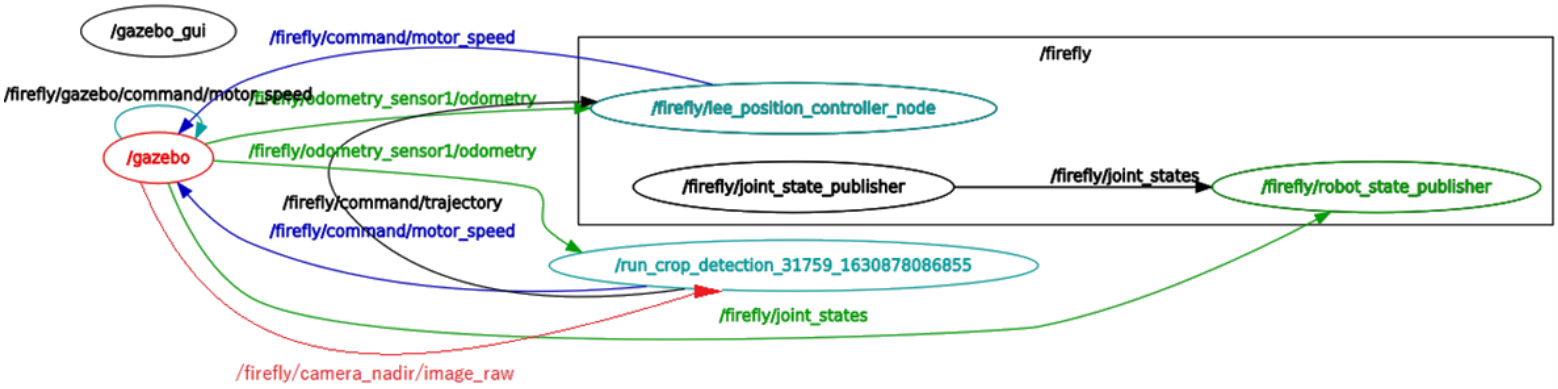


Figura 5. Diagrama de relación entre nodos y tópicos de la simulación.

6. De la detección de cultivos

Se describirá el proceso de detección de cultivos mediante el cual se realizaron pruebas de diferentes métodos de procesamiento de imagen y se emplearon estrategias de identificación de elementos del cultivo haciendo uso de inteligencia artificial, para estos procedimientos se explicará qué elementos fueron utilizados, cuál fue el procedimiento y se indicarán los pros y contras de cada uno de estos. Así mismo se dará a entender las soluciones escogidas y la razón por la cual se optó por estas.

Para el proceso de detección de cultivos se partió de imágenes de cultivos tomadas mediante la cámara Zed en los espacios del campus de la Universidad Militar Nueva Granada. En principio se realizaron tomas frontales del cultivo para luego realizar tomas superiores, esto debido a que con las tomas frontales no se tiene una correcta perspectiva de los objetos y plantas detectados, en el momento en el que se desee trasladar la información encontrada a coordenadas X,Y,Z el proceso es más complicado debido al ángulo, la altura y características de la cámara como distancia focal, para este inconveniente se presenta más adelante transformaciones que se podrían usar si se deseara utilizar estas imágenes. Además de esto, en las tomas frontales se presentan muchos más elementos de ruido y de poca utilidad (edificios, cielo, elementos diferentes al cultivo) que complican el proceso de detección del cultivo.

En los siguientes apartados se presentan los diferentes procedimientos que se realizaron para la detección de cultivos que, si bien finalmente fue escogido uno, cada uno de estos permitió tener una retroalimentación con la cual se corrigieron errores o problemas encontrados durante la implementación del siguiente.

6.1. Segmentación de grises por valor umbral

Para la conversión a escala de grises fueron encontrados dos métodos:

Método de promedio

Mediante el cual se suman los componentes de color de la imagen (R,G,B) y este valor se divide entre 3.

$$Escala\ de\ grises = (R + G + B)/3$$

El problema con este método es que cada color posee una longitud de onda distinta, por lo que al realizar un promedio se asume que cada color contribuye de la misma manera a la imagen, lo cual no es cierto.

Método ponderado o método de luminosidad

Se realiza un cálculo con la suma ponderada de los componentes R,G,B a partir de la recomendación Rec.ITU-R BT.601-7 descrita en la siguiente ecuación:

$$Escala\ de\ grises = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

Este método resuelve el problema del método de promedio, asignándole un peso a cada color dependiendo del aporte que tienen estos en la imagen [50][51].

A partir de imágenes inicialmente frontales tomadas del cultivo, se realizó una conversión de RGB a escala de grises como la presentada en la Figura 6 mediante el método rgb2gray en Matlab que hace uso del método ponderado.



Figura 6. Cultivo en escala de grises.

Con la conversión de esta imagen es posible definir un valor umbral en el que se puede segmentar aquellas zonas de la imagen que hacen referencia al cultivo. Dependiendo de los valores máximos y mínimos de la imagen es posible establecer el valor umbral, a aquellos píxeles con un valor por debajo del valor umbral se les asigna un valor de 0, en caso contrario se les asigna un valor de 255. Realizando este proceso es posible obtener una imagen como la que muestra la Figura 7.



Figura 7. Segmentación de imagen Cultivo en escala de grises a partir de valor umbral.

En esta figura, se observa como aquellos píxeles en blanco, en su mayoría hacen referencia a las zonas verdes del cultivo. Sin embargo, hay casos en los que los reflejos de la luz y otros elementos ajenos al cultivo son identificados en blanco, por lo que este proceso tiene ciertos errores. Es posible variar el valor umbral con el objetivo de reducir el problema de zonas incorrectamente identificadas como cultivo, pero no lo suficiente como para no afectar las zonas correctamente identificadas.

6.2. Segmentación de colores - Matlab

Teniendo en cuenta las desventajas que se presentan al realizar la segmentación a partir de imágenes en escala de grises se buscó la forma de detectar el cultivo sin que la información de elementos ajenos afectara en gran medida. Por esta razón se hizo la segmentación por colores. Mediante la APP de Matlab "Color Thresholder" Se determinó el espacio de color, para el cual fue escogido HSV debido a que experimentalmente fue más sencillo determinar los rangos de valores para la segmentación.

A partir de esta segmentación se obtuvo la máscara que fue aplicada en la imagen para determinar que es cultivo y que no, en el resto de las imágenes. Los rangos de valores escogidos son válidos para diferentes tonalidades de verde observadas en las imágenes de prueba. Se presenta la imagen original y la máscara del proceso de segmentación de color en las Figuras 8 y 9 respectivamente.



Figura 8. Toma original de cultivos en la Universidad Militar Nueva Granada.

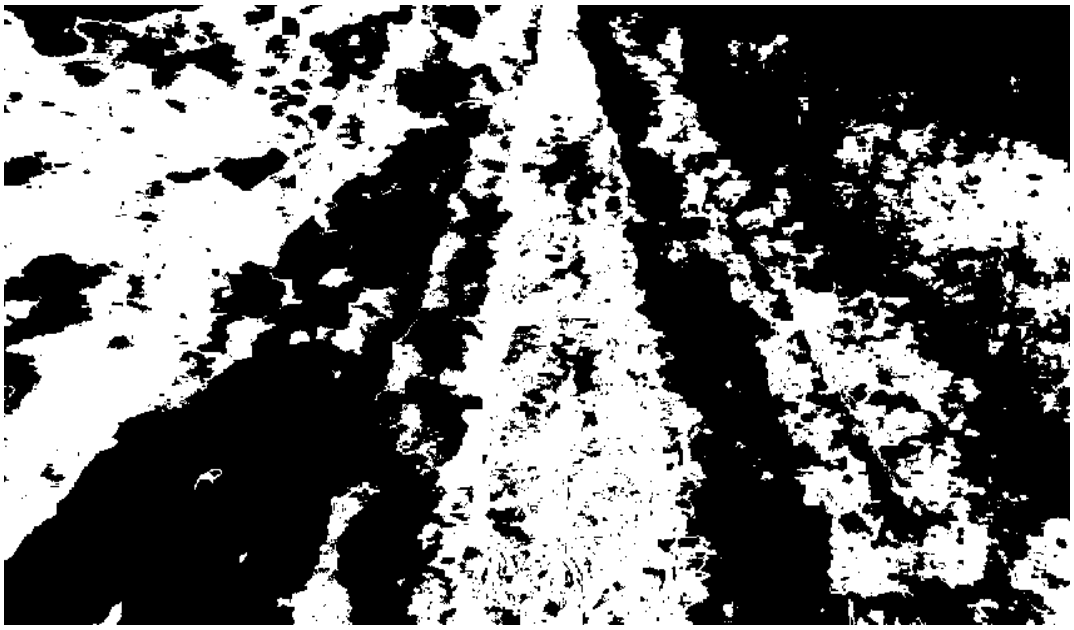


Figura 9. Máscara obtenida de la segmentación de colores.

Y a partir de esta, es posible visualizar en la Figura 10 las regiones de verde de la imagen original al multiplicar la máscara por la imagen del cultivo.

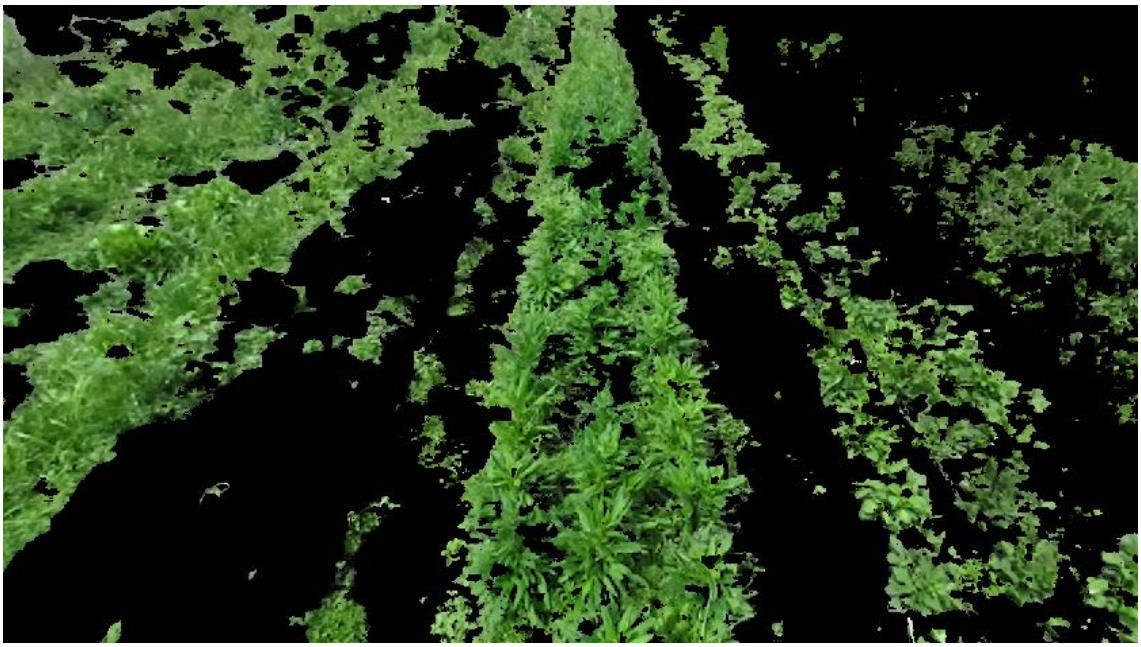


Figura 10. Regiones de verde en el cultivo a partir de máscara de segmentación por color.

6.3. Operaciones morfológicas

Las operaciones morfológicas son un conjunto de operaciones aplicadas a imágenes que realizan un procesamiento basado en formas (elementos estructurales). Para estas operaciones el valor de cada píxel depende de los píxeles vecinos, durante el procesamiento de las imágenes se evaluaron las operaciones de erosión y dilatación [52].

6.3.1. Erosión

La erosión se encarga de encoger o adelgazar los elementos en una imagen. La definición matemática de la operación de erosión de A (la imagen binaria), debido a B (el elemento estructural) es definida por:

$$A \ominus B = \{z|(B)_z \cap A^c \neq \emptyset\}$$

Lo que quiere decir que la erosión de A por B es el set de todas las ubicaciones del origen del elemento estructural en donde al trasladar B, este no se superpone con el fondo de A [53].

En la Figura 12 se describe el proceso en el que se aplica la operación de erosión usando la imagen y el elemento estructural de la Figura 11.

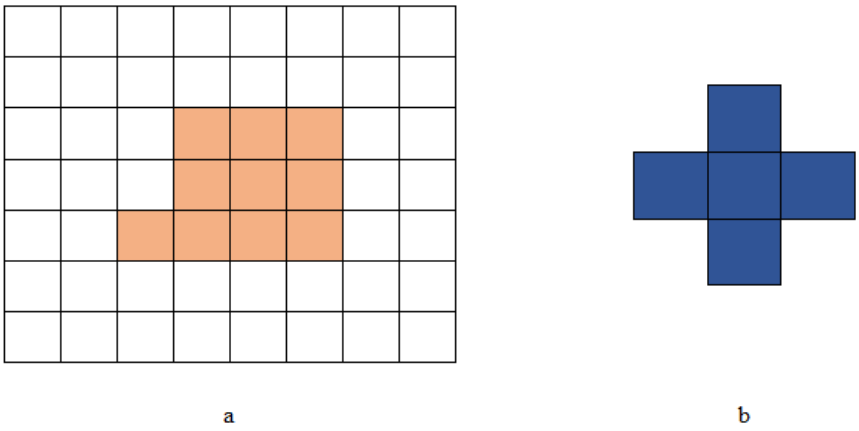


Figura 11. Imagen y elemento estructural utilizados como ejemplo para realizar operaciones morfológicas. a) Imagen Original. b) Elemento estructural.

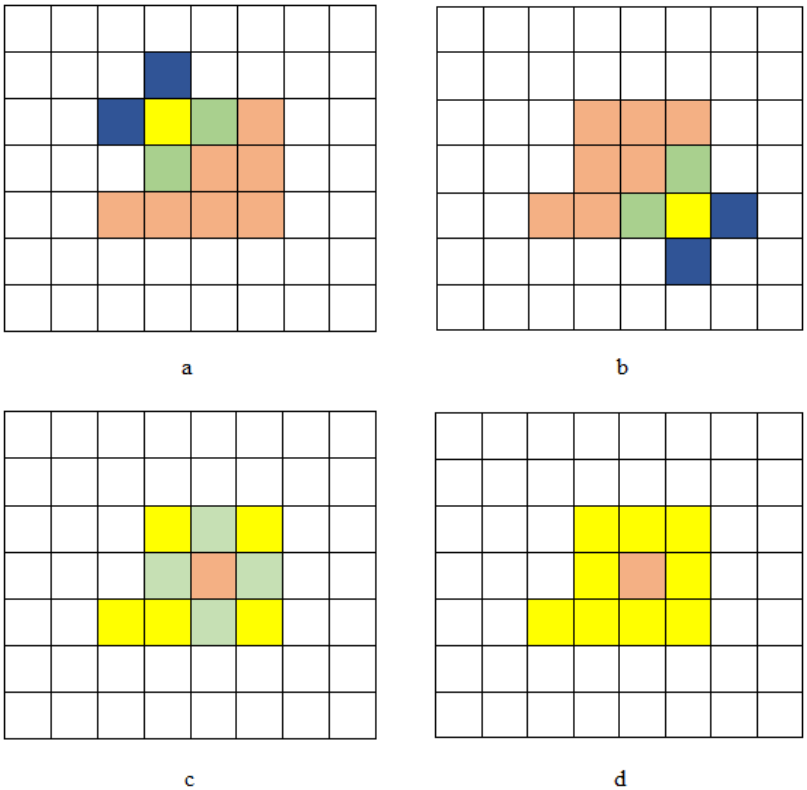


Figura 12. Operación morfológica de erosión aplicada a imagen. a) y b) En verde, superposición entre elemento estructural e imagen; en amarillo, píxel eliminado parte de la imagen a partir del centro del elemento estructural. c) Superposición de

todo el elemento estructural con parte de la imagen, por lo que el centro (en naranja) no se elimina. d) Imagen luego de operación morfológica.

Para la erosión de la imagen binaria del cultivo se escogió como elemento estructural un rectángulo que permitiera eliminar aquellos elementos de verde que se encuentran en los surcos que no contribuyen en la identificación del cultivo, sino que actúan como ruido para el mismo.

Se presenta en la Figura 13 el resultado de aplicar esta operación a la imagen binaria obtenida de la segmentación por color.



Figura 13. Erosión de imagen usando rectángulo como elemento estructural.

Esta operación morfológica se usó durante las primeras versiones de procesamiento de imagen del proyecto. Para la versión final fue descartada debido a que no se consideró necesaria dados los últimos cambios realizados a la etapa detección del cultivo. Sin embargo, puede ser agregada dentro del código con los parámetros que fueron establecidos en su momento.

6.3.2. Dilatación

Esta operación se encarga de agrandar o engrosar los elementos de una imagen binaria. La dilatación de A por B se define matemáticamente por:

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

La dilatación de A por B es el set que consiste en todas las ubicaciones del origen del elemento estructural en donde al reflejar y trasladar B este superpone al menos una porción de A [53].

En la Figura 14 se describe el proceso en el que se aplica la operación de dilatación usando la imagen y el elemento estructural de la Figura 11.

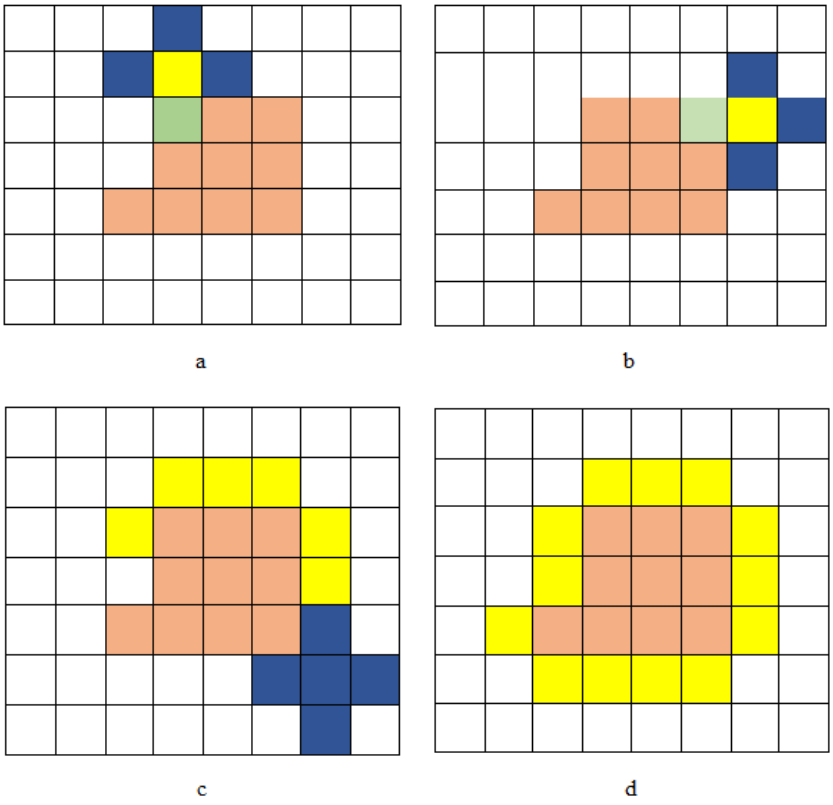


Figura 14. Operación morfológica de dilatación aplicada a imagen. a) y b) En verde, intersección entre elemento estructural e imagen; en amarillo, nuevo píxel parte de la imagen a partir del centro del elemento estructural. c) Resultado luego de no haber interacción entre elemento estructural e imagen. d) Imagen luego de operación morfológica.

Respecto a la dilatación de la imagen fue escogido un rectángulo como elemento estructural. El objetivo de aplicar esta operación fue llenar aquellos espacios y huecos que se encuentran a lo largo del cultivo y que complican la tarea de los algoritmos de distinguir los surcos del cultivo. En la Figura 15 se observa cómo se llenan los espacios en negro que corresponden a la línea de cultivo pero que no son identificados debido a que no son elementos verdes del cultivo.

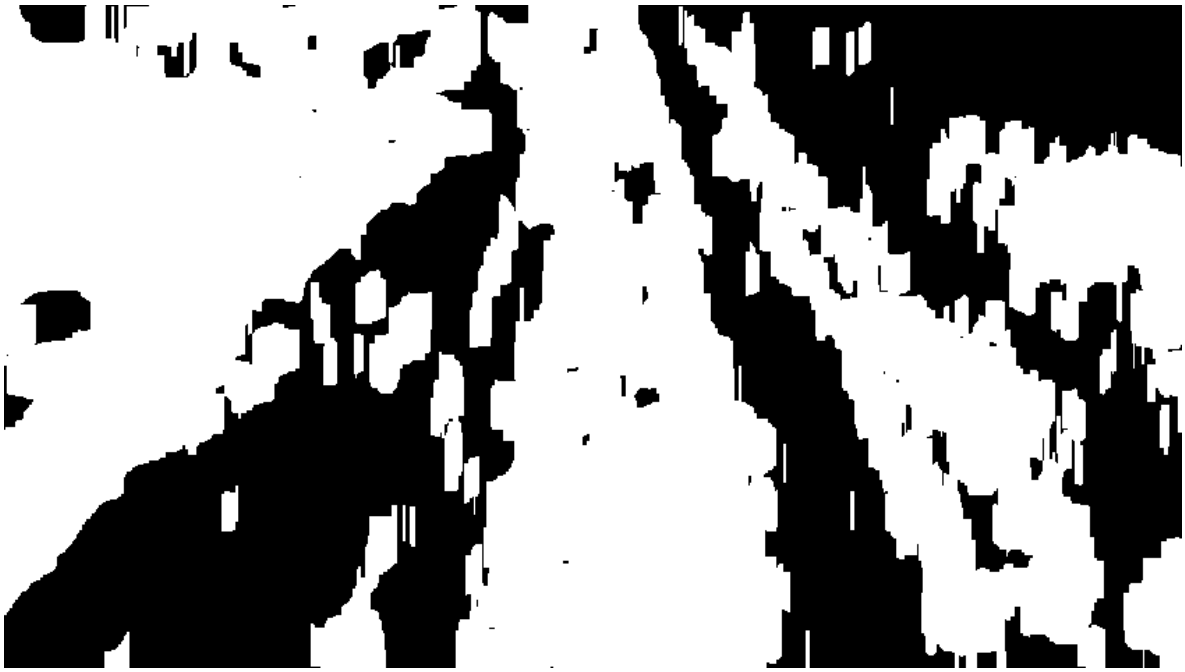


Figura 15. Dilatación de imagen usando rectángulo como elemento estructural.

6.4. Filtro de promedio

Este filtro toma para cada píxel el valor del promedio que hay entre este y sus vecinos. Se realiza el cálculo de la media de los valores de píxeles dentro de un kernel $n\ x\ n$ y con este valor se reemplaza la intensidad del píxel del centro del elemento [54].

A partir de esto es posible llenar aún más los espacios existentes en los cultivos, difuminar elementos ajenos a la línea de cultivo y tener las bases para limitar cada línea de cultivo de

forma más suave y definida. En la Figura 16 se presenta el resultado de usar este filtro, en éste se observa que la imagen no es binaria, sino que se encuentra en una escala de grises.

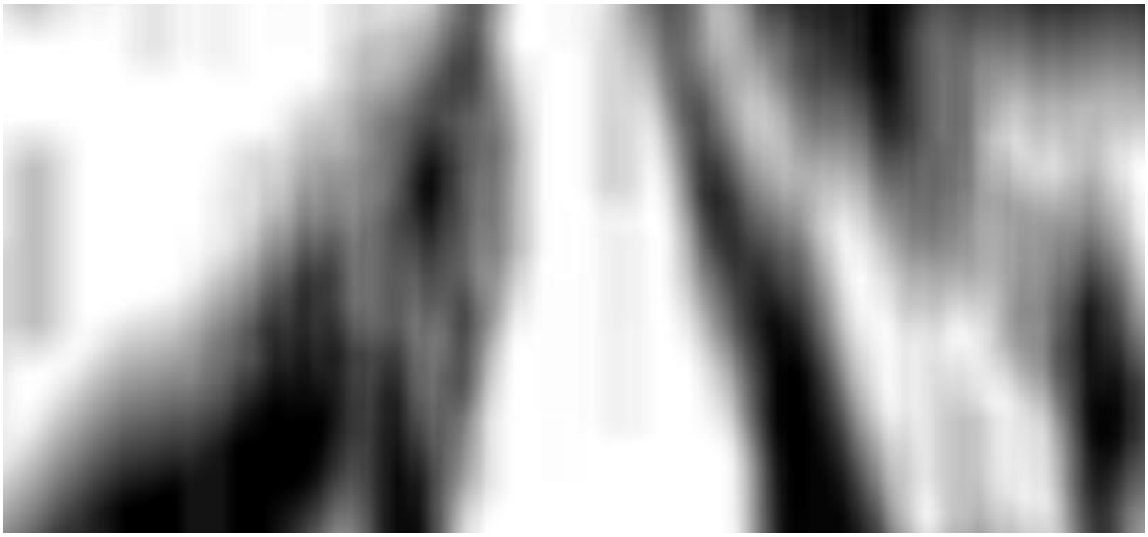


Figura 16. Filtro de promediado de imagen.

6.5. Segmentación por valor umbral II

A partir de la imagen generada por el filtro de promedio, se requiere de nuevo obtener una imagen con valores binarios para definir claramente los límites de cada línea de cultivo, y eliminar aquellos conjuntos de verdes que no pertenecen al cultivo, por lo que se realiza una segunda segmentación a partir de otro valor umbral. El resultado se presenta en la Figura 17.



Figura 17. Segmentación posterior a filtro de promedio.

Dependiendo del valor umbral es posible tener líneas de cultivo más o menos anchas, por esta razón mediante pruebas experimentales se debió escoger un valor de forma tal que las zonas entre cultivos (surcos) no se vieran afectadas y pudieran ser delimitadas claramente.

7. Las texturas y cultivos con bolsas de plástico

Como se pudo observar en las imágenes del cultivo previamente presentadas, los cultivos no son de un solo tipo, hay variedad entre las diferentes plantas que están, y no solamente se puede decir que hay variedad en las plantas, hay otros elementos como la tierra, las bolsas y los obstáculos. Dentro del problema de la detección del surco, también se debe tener en cuenta que hay obstáculos en el cultivo que deben ser identificados y bolsas de plástico en donde también crecen plantas.

Se debe entender entonces que para llevar a cabo el procedimiento de detección de surcos en el cultivo debe haber un entendimiento mayor de lo que hay en las imágenes, no solamente distinguir entre verde y no verde, y para esto hay otras propiedades en la imagen de las que se puede hacer uso, una de estas es la textura.

7.1. Segmentación de textura por filtro de Gabor

El filtro de Gabor es un filtro lineal usado para análisis de texturas, extracción de características, detección de bordes, entre otros. Es un tipo de filtro pasa banda que puede ser interpretado como una señal senoidal modulada por una gaussiana, con un tipo específico de frecuencia y una determinada orientación. Al hacer uso de este filtro, o varios filtros de Gabor con diferentes parámetros cada uno, es posible extraer ciertas características a cada imagen [55].

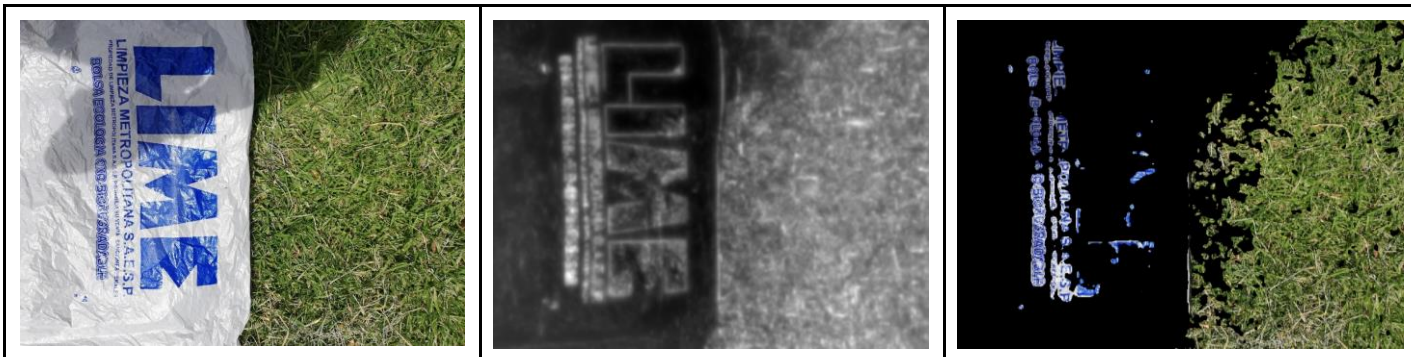
Este filtro se define por la siguiente expresión:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \exp\left[j\omega_x(xcos\theta + ysin\theta)\right]$$

Donde ω_x es la frecuencia de una onda sinusoidal, θ es la orientación del filtro y σ es la desviación estándar de la función Gaussiana en las direcciones x, y [56].

Se presenta en la Figura 18 los resultados de aplicar el filtro de Gabor a dos texturas distintas, en donde previamente la imagen fue convertida a escala de grises.

Se observa que el filtro de Gabor da como resultado una imagen en donde las dos texturas se logran diferenciar a partir de la intensidad de la imagen. La textura del pasto es más clara que la textura de la bolsa, por lo que si se realiza una segmentación a partir de un valor umbral es posible diferenciar las dos texturas.



(a) Imagen de textura. *(b) Imagen con filtro aplicado.* *(c) Segmentación a partir de filtro.*

Figura 18. Filtro de Gabor aplicado en imagen para segmentación texturas de pasto y bolsa.

7.2. Segmentación de textura por filtro de entropía

Mediante este filtro se obtiene una imagen en la que aquellos píxeles que son más brillantes corresponden a aquellos que en el "vecindario" de píxeles poseen una mayor entropía. La entropía es una medida estadística de aleatoriedad, a partir de esta es posible diferenciar entre ciertas texturas presentes en una imagen.

La forma en que Matlab implementa este filtro consta de dos pasos principales:

1. Obtener el histograma de la imagen: Se hace mediante la función *imhist*. A esta función se le entrega una imagen en escala de grises, y devuelve dos arrays: "cuentas" y las ubicaciones de los "bins". Son los datos que representan la cantidad de píxeles para cada valor de intensidad presente en la imagen, y dependen del tipo de imagen [57].

2. Obtener valor de entropía: Entropía que se calcula a partir del histograma de la imagen. Este valor se calcula con la expresión:

$$-sum(p.* log_2(p))$$

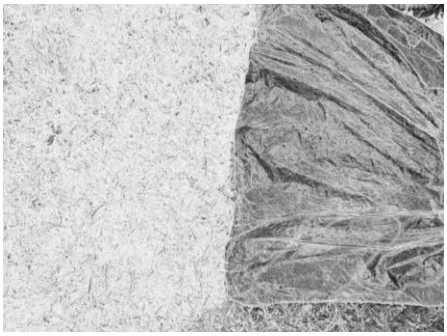
En donde p son las "cuentas" normalizadas retornadas por el histograma de la imagen.

La función entropyfilt retorna un array en donde cada elemento contiene el valor de entropía calculado para un vecindario de $9\ x\ 9$ de cada píxel de la imagen de entrada [58].

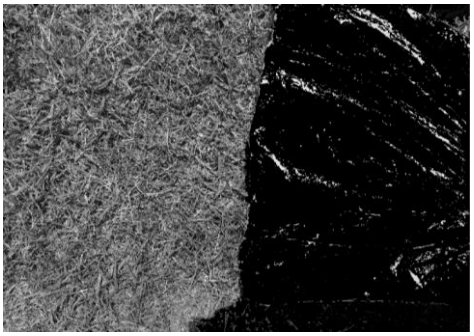
Como se observa, los píxeles relacionados a la textura del pasto tienen una mayor entropía que los de la bolsa, y es posible segmentarlos de acuerdo con un valor umbral. Sin embargo, también se presentan ciertos errores en la diferenciación entre una textura y otra en el caso de una sección del pasto que tiene más sombra que la otra, esta sección es tomada como una textura diferente. La segmentación se presenta en la Figura 19.



(a) Imagen de textura.



(b) Imagen con filtro aplicado.



(c) Segmentación a partir de filtro, en escala de grises.

Figura 19. Filtro de entropía aplicado en imagen para segmentación texturas de pasto y bolsa.

7.3. Matriz de co-ocurrencia GLCM

Es un método estadístico de segundo orden que considera la relación espacial entre píxeles. Esta matriz se crea al calcular la frecuencia con la que se presenta en una imagen, una combinación específica de valores y relaciones espaciales para un par de píxeles [59]. La forma en que se determina el par de píxeles es mediante dos parámetros, distancia y ángulo.

Las relaciones entre píxeles se pueden describir de la siguiente forma: (número de píxeles en la dirección X, número de píxeles en la dirección Y) por ejemplo, (1,0) indica que se tomará el píxel a la derecha del píxel de referencia [60]. De esta forma se pueden definir los ángulos principales para una distancia de un píxel:

(1,0) – Ángulo de 0°

(0,-1) – Ángulo de 90°

(1,1) – Ángulo de 45°

(1,-1) – Ángulo de 135°

Para generar la matriz de co-occurrencia, se establece primero una matriz de NxN en la que N es el número de niveles de gris presentes en la imagen a analizar, y se construye a medida que se desplaza el píxel de referencia a lo largo de la imagen, tomando el número de veces

en las que coinciden los valores de cada par de píxeles seleccionado denominado ventana (píxel de referencia y píxel vecino).

De esta forma, al elegir por ejemplo una relación espacial (1,0) se puede generar la matriz de co-ocurrencia en la Figura 21 para la imagen de 4 niveles de gris en la Figura 20.

0	0	1	1
1	0	0	2
3	3	2	2
1	0	1	0

Figura 20. Imagen para generar matriz de co-ocurrencia.

Píxel de referencia \Píxel vecino	0	1	2	3
0	2	2	1	0
1	3	1	0	0
2	0	0	1	0
3	0	0	1	1

Figura 21. Matriz de co-ocurrencia (encerrada en gris) generada a partir de la Figura con una relación espacial de (1,0).

Para poder realizar los cálculos respectivos se requiere obtener la matriz simétrica de esta matriz de co-ocurrencia, lo que se puede obtener contando cada par de píxeles dos veces, una hacia la derecha (1,0) y otra hacia la izquierda (-1,0), resultado que se puede observar en la Figura 22.

4	5	1	0
5	2	0	0
1	0	2	1
0	0	1	2

Figura 22. Matríz de co-ocurrencia simétrica

Finalmente, esta matriz debe ser expresada como probabilidad, para esto se lleva a cabo un proceso de normalización en el que se divide cada uno de los valores entre todas las veces posibles que puede suceder la combinación de cada par de nivel de gris, que en este caso sería de 24 (12 hacia la izquierda y 12 hacia la derecha) [61].

Esto se representa con la siguiente expresión:

$$C_{i,j} = \frac{V_{i,j}}{\sum_{i,j=0}^{N-1} V_{i,j}}$$

Donde:

- i es el número de filas
- V el valor de la celda (i,j) en la ventana
- $C_{i,j}$ es la probabilidad en la celda i,j
- N es el número de filas o columnas

El resultado se presenta en la Figura 20.

0.166	0.208	0.042	0
0.208	0.083	0	0
0.042	0	0.083	0.042
0	0	0.042	0.083

Figura 23. Matriz de co-ocurrencia normalizada.

A partir de esta matriz es posible extraer medidas estadísticas de la imagen descritas en [60] y [61], presentadas a continuación:

- **Contraste:** Es una medida de la variación local en una imagen, a mayor contraste mayor es el valor, el cual se incrementa exponencialmente.

$$\sum_{i,j} (i-j)^2 \cdot C_{i,j}$$

- **Disimilitud:** Similar al contraste, cuando los valores son bajos representan píxeles similares a sus vecinos, a medida que aumentan quiere decir que son diferentes. A diferencia del contraste este valor aumenta de forma lineal.

$$\sum_{i,j=0}^{N-1} C_{i,j}|i-j|$$

- **Homogeneidad:** Medida opuesta al contraste.

$$\sum_{i,j} \frac{C_{i,j}}{1+(i-j)^2}$$

- **ASM:** Es una medida de la homogeneidad local. Sus valores son altos cuando en la matriz de co-ocurrencia hay pocas entradas de gran magnitud, y son bajos cuando los valores son similares.

$$\sum_{i,j}^{N-1} {C_{i,j}}^2$$

- **Entropía:** Mide la aleatoriedad de la imagen, su valor es mayor entre más similares sean los elementos de la matriz de co-ocurrencia.

$$\sum_{i,j} -C_{i,j} \cdot log_2[C_{i,j}]$$

7.4. División de imagen y clasificación de texturas

Para llevar a cabo el cálculo de la matriz de co-ocurrencia haciendo uso de las imágenes previamente trabajadas se requiere dividirlas en pequeñas secciones en las cuales se pueda diferenciar cada una de las texturas existentes.

Para este trabajo se obtuvo la matriz de co-ocurrencia a partir de la cual se realizó el cálculo de las propiedades de correlación, disimilitud, contraste, homogeneidad y ASM.

Mediante el cálculo de estas propiedades es posible diferenciar entre distintas texturas, para evidenciar esta diferencia es posible realizar una gráfica 2D de imágenes con las que se realizó la prueba teniendo en cada eje una de las propiedades, esto se presenta en la Figura 24, para la cual se generó una matriz de co-ocurrencia con valores de distancia de 1 píxel y ángulo de $\frac{\pi}{2}\ rad$.

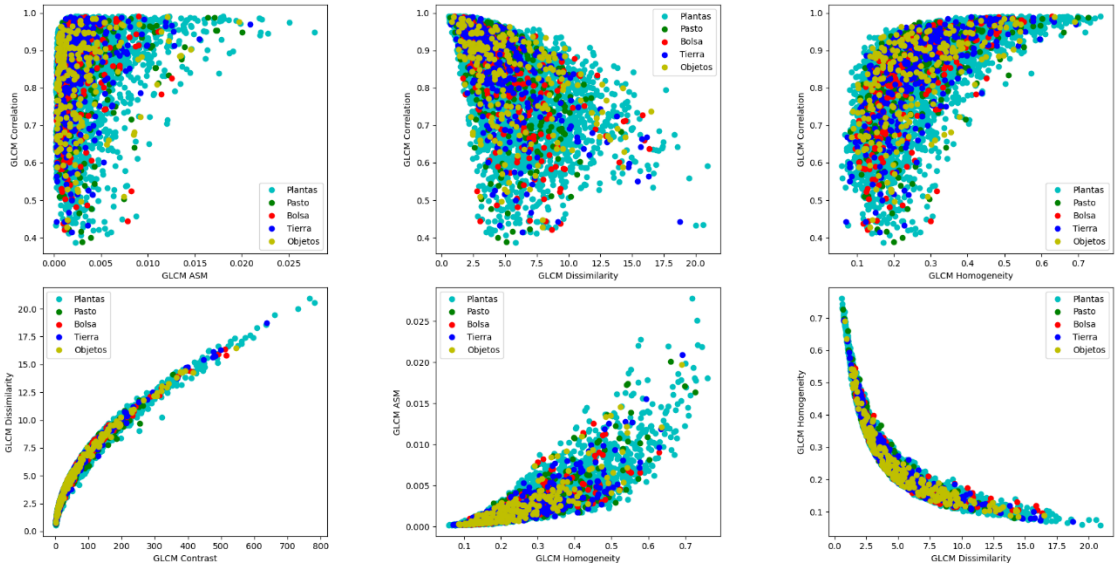


Figura 24. Gráfica de propiedades obtenidas a partir de la matriz de Co-Ocurrencia.

7.5. Red Neuronal para propiedades de imagen – matriz de co-ocurrencia

A plena vista no es fácil diferenciar estas propiedades debido a que si se quisiera graficar las diferentes propiedades se trabajaría con un plano de 5 dimensiones, por este motivo se buscó un método que permitiera diferenciar entre cada uno de los grupos (texturas) existentes. Para lo cual se escogió hacer uso de una red neuronal de tipo perceptrón multicapa, cuyas características permiten aplicarla para esta clase de problemas. Esta red fue diseñada y entrenada en Tensorflow.

En este caso, se realizó el cálculo de la matriz de co-ocurrencia, con el que fue posible extraer cada una de las características escogidas previamente, y se usó cada una de estas características como entradas a la red neuronal. De esta forma, por cada textura se obtienen características que permiten clasificarla dentro de los diferentes tipos de textura establecidas.

Para que funcione correctamente se requeriría de imágenes más precisas con respecto a la textura que se maneja, debido a que las imágenes que se lograron tomar como prueba en general contienen más de una textura. Si se quisiera tener imágenes con texturas más claras habría que dividir la imagen en partes más pequeñas consiguiendo una mayor “definición”.

Después de diferentes pruebas en las que se modificaron los hiperparámetros, se seleccionó la mejor arquitectura para la red con los hiperparámetros presentados en la Tabla 1, que permitieron obtener una mayor precisión.

Capa	forma de la salida	Número de parámetros
dense_1	(None, 512)	3072
activation_1	(None, 512)	0
dropout_1	(None, 512)	0
dense_2	(None, 256)	131328
activation_2	(None, 256)	0
dropout_2	(None, 256)	0
dense_3	(None, 128)	32896
activation_3	(None, 128)	0
dropout_3	(None, 128)	0
dense_4	(None, 5)	645
activation_4	(None, 5)	0
Total parámetros entrenables		167941

Tabla 1. Hiperparámetros de red neuronal.

En la Tabla 2 se presenta el valor de precisión de la red neuronal, junto con el tiempo que tarda el cálculo de las propiedades de co-ocurrencia para diferentes ejecuciones de la misma imagen. Si bien el tiempo de cálculo de las propiedades es aceptable para la aplicación que se le dará a la red, el valor de precisión alcanzado con la red no es suficiente para una correcta detección de las texturas. Aunque el caso de que la red confunda la clase “pasto” con la clase “planta” no es algo grave, dado que la precisión es baja existe posibilidad de confundir una bolsa con un surco, lo cual puede afectar bastante la información que se espera recibir sobre el cultivo y hacer que los límites se definan incorrectamente haciendo que se presenten errores o resultados ineficientes en el cálculo de la trayectoria.

Dato	Valor
Precisión de validación	0.7076

Tiempo de cálculo de propiedades para matriz de co-ocurrencia (para una sola imagen) en primera ejecución	0.0638s
Tiempo de cálculo de propiedades para matriz de co-ocurrencia (para una sola imagen) en segunda ejecución	0.0079s
Tiempo de cálculo de propiedades para matriz de co-ocurrencia (para una sola imagen) en tercera ejecución	0.0199s

Tabla 2. Resultados obtenidos al usar matriz de co-ocurrencia y red neuronal.

7.6. Redes convolucionales:

Una red neuronal convolucional es una variación de la red trabajada en el apartado anterior, un perceptrón multicapa. En este caso en lugar de ingresarle con un array de datos, se entrega una imagen que es procesada por una red neuronal en Tensorflow con la que se clasifican las imágenes según las diferentes clases existentes: pasto, planta, bolsa, tierra y obstáculos.

La arquitectura de la red se presenta en la Tabla 3.

Capa	Forma de la salida	Número de parámetros
conv2d_1 (Conv2D)	(None, 222, 222, 32)	896
activation_1 (Activation)	(None, 222, 222, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_2 (Conv2D)	(None, 109, 109, 64)	18496
activation_2 (Activation)	(None, 109, 109, 64)	0

max_pooling2d_2 (MaxPooling2D)	(None, 54, 54, 64)	0
flatten_1 (Flatten)	(None, 186624)	0
dense_1 (Dense)	(None, 64)	11944000
activation_3 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 5)	325
activation_4 (Activation)	(None, 5)	0
Total parámetros entrenables		11963717

Tabla 3. Arquitectura de red convolucional.

En la Tabla 4 se presenta el valor de precisión de la red neuronal, junto con el tiempo que tarda la primera y segunda ejecución de la red para las mismas 250 imágenes.

Dato	Valor
Precisión de validación	0.8763
Tiempo de primera ejecución de red neuronal para 250 imágenes.	1.9169s
Tiempo de segunda ejecución de red neuronal para 250 imágenes.	0.790s

Tabla 4. Resultados obtenidos al usar red convolucional.

A pesar de que es posible diferenciar las texturas aplicando estos filtros, también se debe tener en cuenta que, aunque en su mayoría es posible distinguir una textura de la otra, la intensidad de la luz afecta el resultado de esta segmentación, y en condiciones de luz desconocidas para un cultivo, se pueden llegar a evidenciar algunos errores. Además de que las bolsas y el pasto no serán las únicas texturas por analizar, como ya se ha mencionado hay bastante variedad de texturas y si estas no se generalizan en determinados grupos habría que

realizarse una segmentación específica para cada una de ellas, lo que implicaría conocer previamente el cultivo que se fuera a trabajar.

Todas estas condiciones se tuvieron en cuenta al momento de preparar el *dataset* con el que sería entrenada la red neuronal, se buscó generalizar las diferentes texturas del cultivo de manera en que no fuera necesario tener una etiqueta por cada tipo de planta, pero que también existieran las suficientes etiquetas para texturas que son bastante diferentes o que son importantes de diferenciar. El entrenamiento de la red se realizó variando el ángulo de las imágenes, su brillo y su escala, dado que cuando se aplique la red neuronal a las imágenes que toma el dron en tiempo real, este se va a encontrar en movimiento y como ya fue mencionado, las condiciones de luz varían dependiendo de factores naturales que no se pueden controlar.

Se presenta en la Figura 25 un ejemplo (a una altura mucho mayor a la que trabajaría el dron) de cómo se dividen los fotogramas del video que captura el dron y cómo son clasificados por la red neuronal.

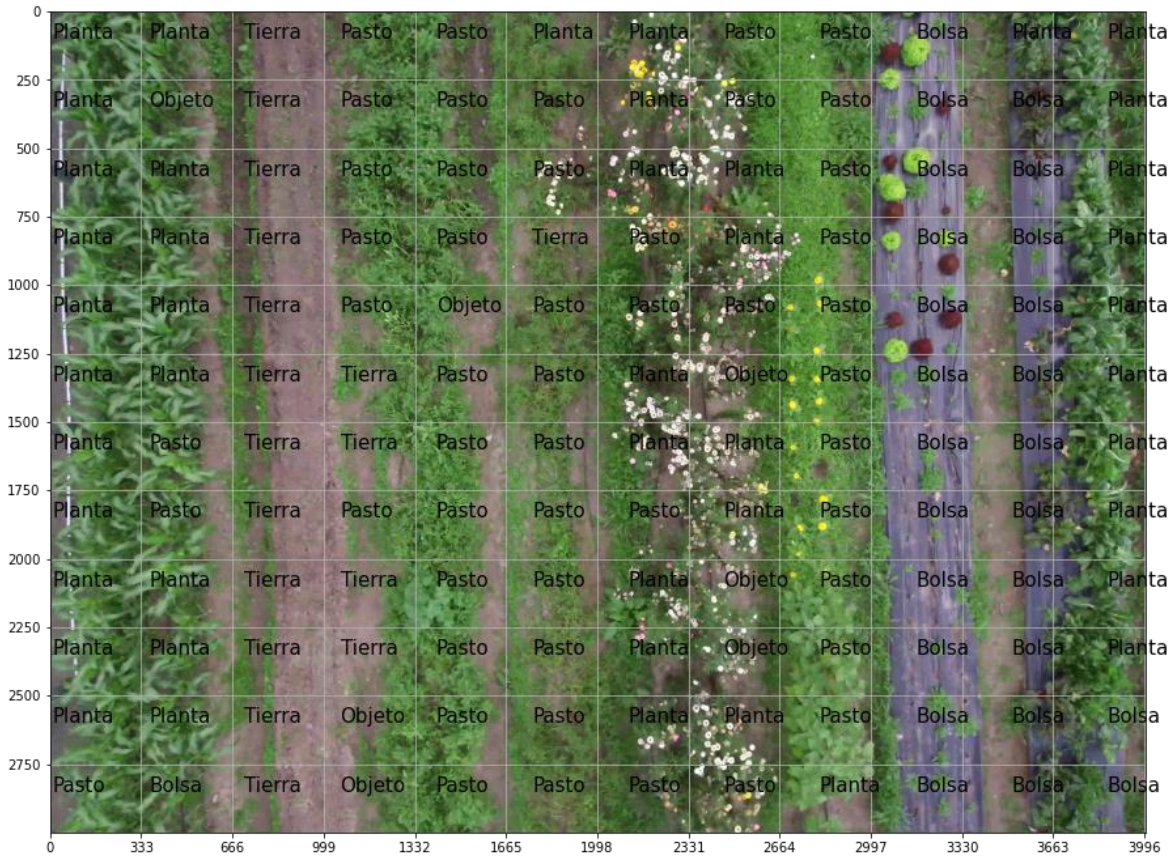


Figura 25. División de imagen en celdas para uso de red convolucional.

Es posible observar que en su mayoría el dato obtenido para las celdas coincide con lo que significa cada una de ellas, en pocos casos se confunde “Pasto” con “Planta”, sin embargo, esta clase de errores no se diferencian al momento de distinguir el cultivo de los surcos, debido a que “Pasto” es otra forma de identificar el cultivo, pero su textura fue tomada como distinta dentro del proceso de clasificación.

7.7. Resultado final del proceso de detección de cultivos desde Gazebo:

Luego de efectuar todas las configuraciones requeridas para el dron, fue posible capturar las imágenes del cultivo, evaluar que operaciones sobre la imagen se debían usar (verificando según su impacto en la imagen si son requeridas o no), y llevar a cabo todo el proceso de detección del cultivo e identificación de límites y centros de este.

Los pasos en este proceso son los siguientes:

- Segmentación por colores
- Filtro de promedio
- Segmentación de grises por valor umbral II
- Dilatación de imagen
- Red Neuronal Convolucional
- Detección de límites de cultivo
- Detección de centros de cultivo

Mientras que la segmentación por colores y las operaciones morfológicas permitieron diferenciar parte de las zonas verdes del cultivo, la Red Neuronal Convolucional identificó aquellas zonas del cultivo que tenían colores distintos, los cultivos que se encontraban sobre bolsas plásticas y finalmente los objetos que se detectaron los cuales no pertenecen ni al cultivo ni a los surcos. Se presenta en la Figura 26 el resultado de este proceso:



(a) Imagen original



(b) Segmentación a partir de procesamiento de imagen y detección de red convolucional

Figura 26. Segmentación de imagen luego de pasar por detección de elementos mediante red convolucional.

Si bien la tarea de identificación de cultivos se podría haber realizado teniendo únicamente una red convolucional, se evidenció durante las pruebas que los procedimientos previos permitieron extraer más información de la imagen, de esta forma los inicios y finales de cada línea de cultivo se relacionan más con la imagen capturada. De lo contrario se habrían obtenido segmentaciones rectangulares dadas las características del proceso de división en rejillas, que podrían haber resultado en saltos bruscos entre los desplazamientos que realiza el dron.

8. Delimitación del camino de cultivo

El siguiente paso luego de segmentar aquellos elementos del cultivo es obtener las posiciones en las cuales se encuentran las líneas de cultivo. De esta forma será posible indicarle al dron cuál es el camino que debe recorrer.

8.1. Mediante centroides de agrupación de píxeles

Luego de realizar la respectiva segmentación del cultivo se implementó un proceso posterior acorde a la literatura en el cual se dividió la imagen en rejillas de tamaño MxN. Para cada una de las celdas se realizó un promedio del valor existente de sus píxeles, y de nuevo, se evaluaba si este promedio superaba o no un valor umbral. Si el valor umbral se superaba, entonces la celda pertenecía a una zona del cultivo, de lo contrario pertenecía al surco del cultivo.

Este flujo se identifica en el diagrama de la Figura 27.

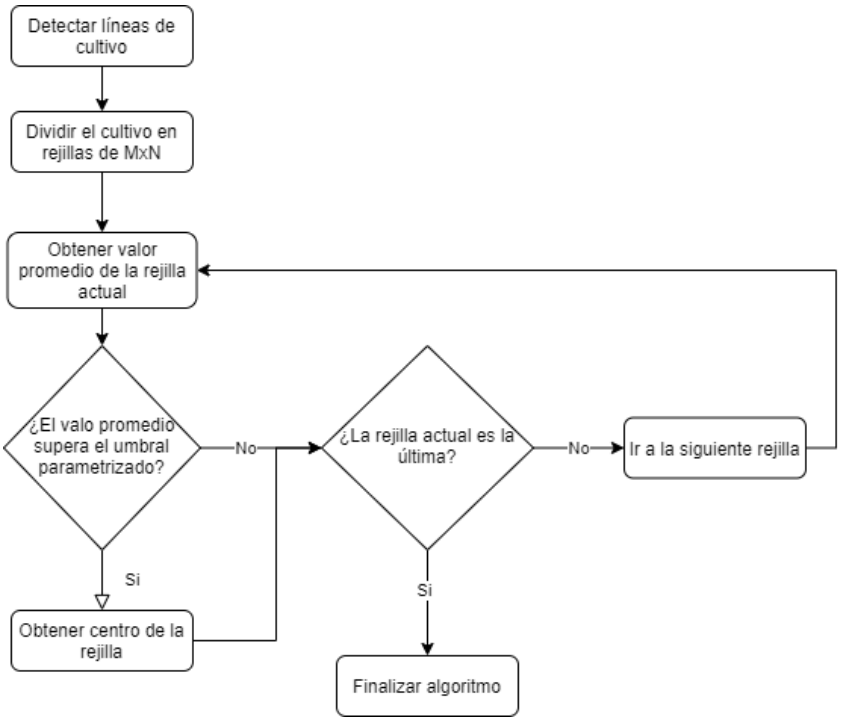


Figura 27. Diagrama del algoritmo de detección de camino de cultivo mediante agrupación de píxeles.

Para aquellas celdas que pertenecían a la zona del cultivo se realizó el cálculo del centroide, que corresponde al píxel central de la celda (en azul). Se presenta en la Figura 28 el resultado de este proceso.



Figura 28. Detección de centroides por acumulación de píxeles.

A partir de los centroides existentes en cada una de las celdas de la imagen se buscó calcular los surcos del cultivo, los cuales se encuentran en medio de dos de los centroides previamente determinados, para la Figura 29 se muestran en naranja los centros de los surcos calculados.

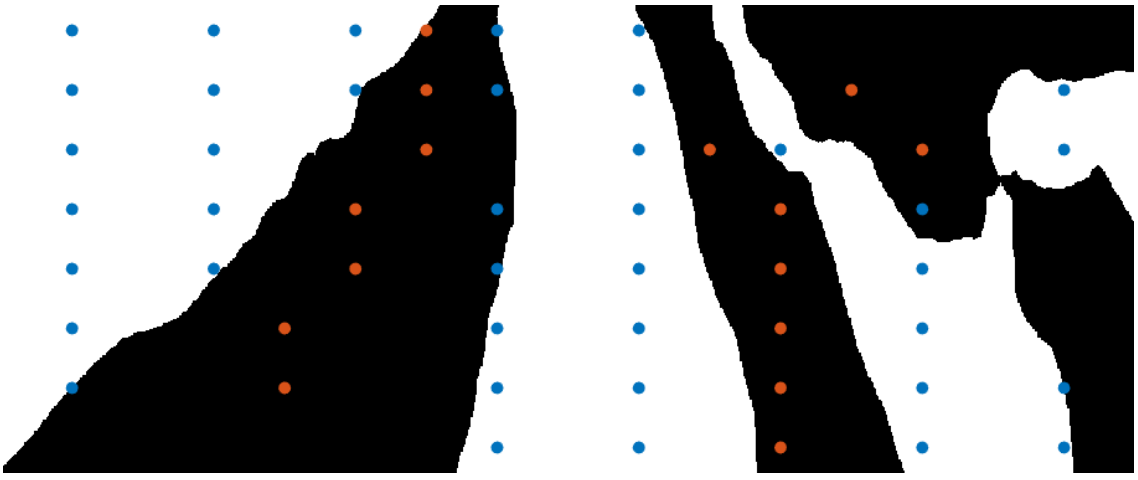


Figura 29. Detección de surcos de cultivo por centroides de acumulación de píxeles.

Como se puede observar, hay puntos en los que el camino o surco no son detectados correctamente debido a que en algunos casos las celdas se solapan. Esto debido a la perspectiva que se tiene; las celdas son del mismo tamaño sin importar que tan cerca o lejos se encuentren de la cámara lo que causa que en algunos casos dos cultivos queden en una misma celda y sea calculado un solo centroide para dos cultivos. Este inconveniente podría ser solucionado si se adecua la rejilla de acuerdo con el ángulo en el que es tomada la imagen, o hacer uso de imágenes tomadas desde la vista superior.

8.2. Detección de límites de cultivo

De acuerdo con los resultados obtenidos por la detección de camino por agrupación de píxeles se decidió mejorar este algoritmo debido a que no es suficientemente precisa la ubicación del centro del camino. Por esta razón se decidió en primer lugar, encontrar los límites de aquellos caminos de cultivo detectados. Esto se hizo realizando un barrido de aquellos píxeles en donde comienza un cultivo (píxeles de color blanco cuyo píxel anterior es de color negro) y donde finaliza un cultivo (píxeles de color negro cuyo píxel anterior es de color blanco). El algoritmo se expone en el diagrama de la Figura 30 y los píxeles mencionados se resaltan en color rojo y color azul respectivamente en la imagen procesada del cultivo de la Figura 31.

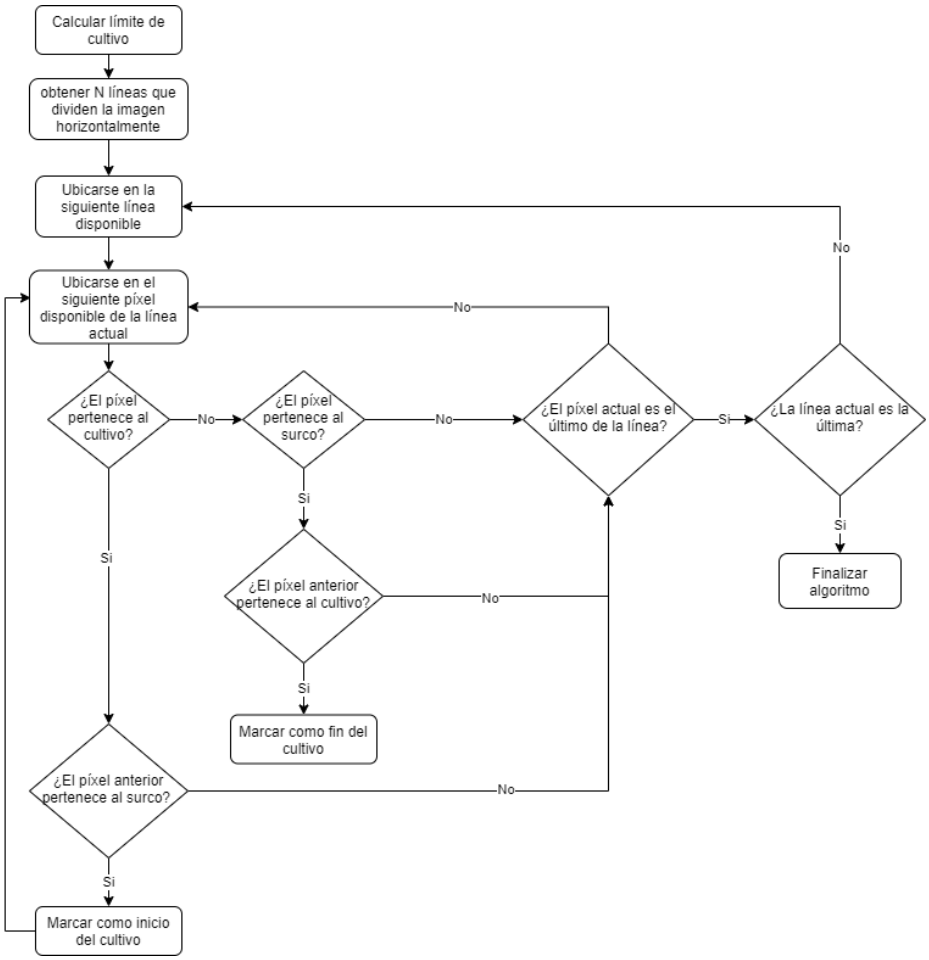


Figura 30. Diagrama del proceso de detección de límites de cultivo.



Figura 31. Límites (inicio y fin) de las líneas de cultivo.

Como se puede observar se delimita correctamente el inicio y final del cultivo, a partir de esto es posible identificar cual es el centro del surco; además de esto es posible parametrizar

cuantos píxeles se quiere detectar respecto al eje Y de la imagen. Claramente esta detección depende bastante de las imágenes tomadas y del procesamiento posterior que se le realice a estas imágenes. Para este algoritmo fue importante tener en cuenta que el ancho del cultivo no es constante, por lo que se definieron límites basados en los píxeles detectados previamente para no afectar la detección del resto de puntos.

8.3. Detección de centros de surco a partir de límites de cultivo

Luego de obtener los píxeles de inicio y fin del cultivo se debe hallar el punto medio a partir de las posiciones de estos. Si bien en un comienzo se detectaron todos los puntos medios para cada surco encontrado dentro del algoritmo, se optó por tener un punto de referencia en el que el dron comenzará el recorrido. A partir de este punto de referencia se define un único surco que será definido a lo largo de la trayectoria, el cual se puede observar en la figura 32.

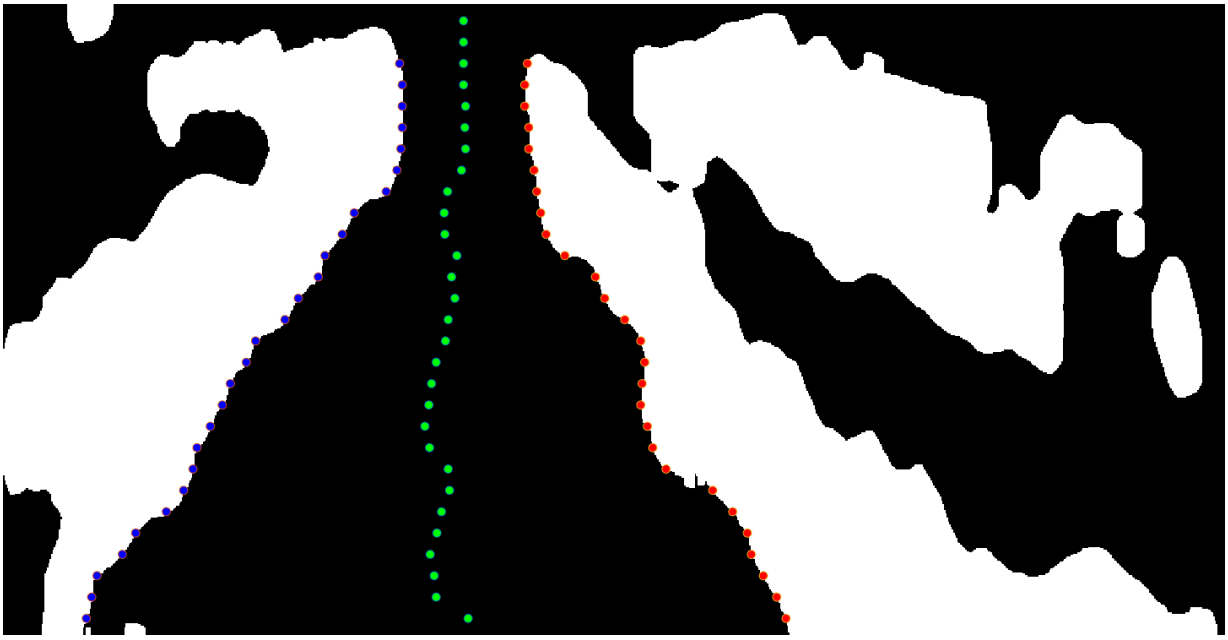


Figura 32. Detección de centro de surcos usando límites de cultivo.

Con el algoritmo realizado ya es posible adaptarlo para tomar cada una de las fotos capturadas por la cámara del dron y realizar el procesamiento de imagen junto con la posterior detección de elementos del cultivo.

Inicialmente durante el proyecto se realizó la detección del centro de los surcos, sin embargo, posteriormente se optó por detectar el centro del camino de cultivo dado que este es el que requiere el robot Ceres_AgroBot para desplazarse. Esto no representa un cambio

significativo dado lo similar que es identificar los centros para estos dos elementos de la imagen binaria.

8.4. El problema de la perspectiva

Como se comentó previamente y con las imágenes evidenciadas, la perspectiva frontal desde la que se toman algunas las imágenes no es la óptima para la detección de los caminos puesto que vuelve más complicada la tarea de diferenciar lo que es cultivo de lo que no en la etapa de procesamiento de imagen dado que para las partes más lejanas del cultivo se comienzan a solapar y a confundir estos datos. Además de esto, se debe tener una relación entre los datos de la imagen detectada y la distancia que hay realmente entre datos.

Por esta razón, se tiene como opción buscar alternativas que solucionen este inconveniente. Una de ellas es convertir las imágenes tomadas desde esa perspectiva, a la perspectiva deseada, otra es realizar las tomas de imagen directamente sobre el cultivo.

8.4.1. Mapeo inverso de perspectiva

Este método consiste en mapear imágenes a un nuevo sistema de coordenadas en el que los efectos de perspectiva son removidos.

Dentro de [62] se hace referencia al método aplicado a imágenes de carreteras y se menciona que la presencia de obstáculos afecta la efectividad del método. Por lo que este sería un punto negativo si se quisiera aplicar para este proyecto en el que no sólo hay obstáculos dentro de cultivo, sino también los datos del cultivo mismo pueden verse afectados por el cambio de perspectiva.

A modo de prueba, se hizo uso de la función **BirdsEyeView** en la que se realiza el proceso de convertir una imagen a una “vista de pájaro” para posteriormente convertir esta imagen a una vista superior relacionada a las coordenadas del vehículo, que en este caso es el dron. Para hacer uso de esta función se deben conocer datos de cómo se realizó la toma de imágenes, por ejemplo, la altura a la que se encontraba la cámara, el ángulo desde el que se tomó, la distancia focal de la cámara, entre otros.

El resultado de este proceso se presenta en la Figura 33 en donde se puede observar que la imagen se distorsiona debido al cambio de perspectiva. También se trató de determinar la distancia real de los puntos detectados.

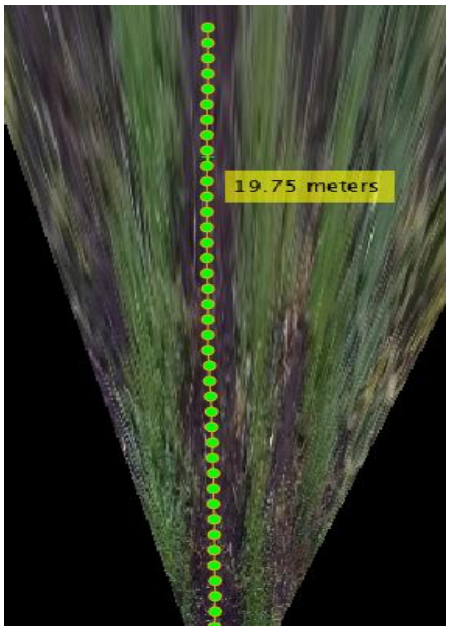


Figura 33. Mapeo inverso de perspectiva aplicado a imagen de cultivo.

Dada la posibilidad de realizar tomas verticales de la imagen se decidió realizarlo de esta manera, de forma tal que las imágenes no fueran distorsionadas y no se tuviera que pasar por un proceso adicional que sumara carga computacional y tiempo al algoritmo.

9. Detección de objetos

En este capítulo se hablará de los algoritmos investigados para detectar aquellos elementos que no pertenecen al cultivo y que se pueden convertir en obstáculos en el recorrido que debe realizar el dron. Se evaluó el resultado de la red convolucional previamente descrita, y el algoritmo YOLOv4 escogido. Se mencionará para este último, el proceso de etiquetado, cómo fue ejecutado y los resultados de la detección en imágenes de prueba.

9.1. Detección de objetos con Red Convolucional

Dentro de los objetivos de este proyecto se encuentra la detección de objetos. Si bien en la simulación tanto los elementos del cultivo como los otros elementos se encuentran en 2D, es importante tener en cuenta aquellos objetos ajenos a las plantas y a los surcos que pueden afectar el desplazamiento del dron en un ambiente real, o que por alguna razón requieren ser encontrados y ubicados dentro del entorno.

Por esta razón también se encuentra la etiqueta de objetos para la red convolucional, para la cual se entrenó con imágenes de ramas, bloques y mangueras que se encontraron en las imágenes de los cultivos de la universidad. Dado que no fueron suficientes imágenes para el entrenamiento, se agregaron otros elementos externos como baldes y botellas, que fueron editados dentro de la imagen del cultivo.

Dentro del código de detección de imagen, se agregó una función que obtuviera la posición actual del dron por medio de un topic y los centros de las rejillas. En dado caso que una de estas rejillas corresponda a un objeto detectado, se hace uso de una función que emplea la equivalencia Píxel – Distancia y la posición actual del dron para determinar la posición del objeto.

A pesar de la precisión obtenida en el entrenamiento, los resultados no fueron los esperados, la detección de objetos no fue precisa. Se realizaron pruebas en la red para entender estos resultados. En la Tabla 5 se presenta la matriz de confusión que evidencia las detecciones correctas, falsos positivos y falsos negativos obtenidos por las pruebas.

	Bolsa	Objeto	Pasto	Planta	Tierra
Bolsa	39	5	0	3	0
Objeto	0	1	1	1	0
Pasto	0	1	23	4	0
Planta	0	4	2	48	0
Tierra	0	15	0	0	40

Tabla 5. Matriz de confusión para red convolucional.

Como se observa, la detección fue correcta en la mayoría de los casos para las clases diferentes a objetos. Teniendo en cuenta que hay una diferencia considerable entre la cantidad de imágenes de las otras clases y la clase objeto, y debido a que para estas otras clases fue mucho más acertada la detección, la precisión general es alta, a pesar de la baja precisión al momento de detectar los objetos.

Analizando estos resultados, se decidió cambiar de estrategia para la detección de objetos. Se conservó la red convolucional debido a que se obtiene información útil del cultivo y de las bolsas sobre las que se trabaja, pero se reemplazó el modelo con el que se detectan los obstáculos.

9.2. Detección de objetos con Modelo YOLO

Se buscó implementar el algoritmo YOLO (*You Only Look Once*) debido a que según las comparaciones realizadas presenta una relación precisión/velocidad más alta que otros algoritmos como Fast RCNN o SSD [43][44], lo cual es bastante útil para la aplicación que se busca realizar, si se tiene en cuenta que además de este hay otro modelo de red neuronal siendo utilizado [63].

Una de las ventajas de usar este algoritmo es que cuando se identifican los objetos, no se requiere realizar un proceso adicional para obtener la posición de este, dado que durante el proceso se generan cajas que encierran al objeto, por lo que no solo se tiene la posición, sino que también el espacio que ocupa el objeto, que es más preciso que las celdas de tamaño fijo con las que se trabajaba con la red convolucional.

El procedimiento realizado para el entrenamiento y la implementación de YOLOv4 se muestra en los diagramas de la Figura 34 y 35 respectivamente, y es descrito posteriormente.

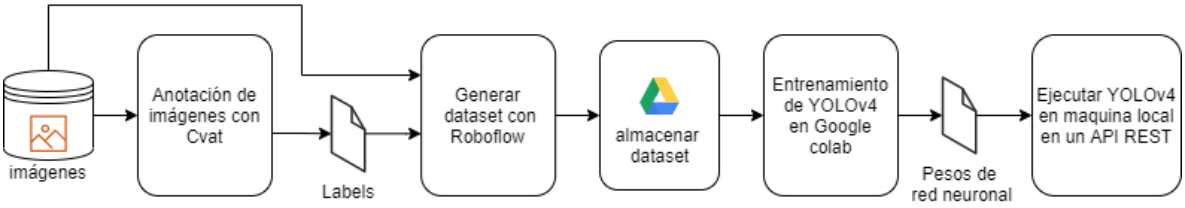


Figura 34. Diagrama del entrenamiento e implementación del algoritmo YOLOv4.

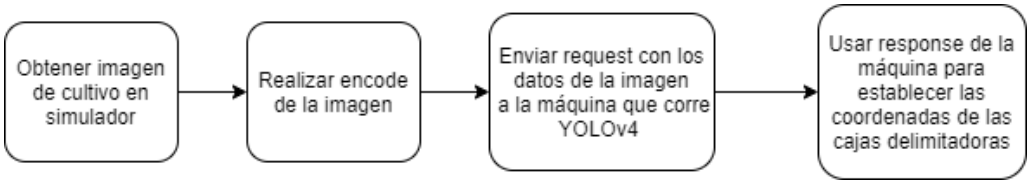


Figura 35. Diagrama del uso de algoritmo YOLOv4 en simulación.

9.2.1. Carga y etiqueta de las clases

Las imágenes base a partir de las que se entrenó el algoritmo son las mismas que se usaron para entrenar la red convolucional, sin embargo, el proceso de generación del *dataset* es bastante distinto. En este caso las imágenes no se dividieron en otras más pequeñas para clasificar cada una de estas, lo que se realizó fue un proceso de anotación de cada uno de los elementos existentes en las imágenes. Esto se realizó con la herramienta *Computer Vision Annotation Tool* (Cvat).

El proceso consistió en cargar las imágenes a la herramienta, crear las etiquetas correspondientes a cada clase que se planea detectar, y posteriormente encerrar cada uno de los elementos identificados con la etiqueta a la que pertenece el elemento. Aunque el objetivo de usar este algoritmo es solamente detectar los objetos, es necesario identificar otras clases para que este pueda diferenciar correctamente entre plantas con formas extrañas, bolsas del cultivo, y los objetos. Este etiquetado se presenta en la Figura 36.



Figura 36. Etiquetado de clases en imágenes. Amarillos: Objetos, Verdes: plantas, Violeta: bolsas.

9.2.2. Dataset de entrenamiento

Hasta el punto anterior se tiene etiquetadas las imágenes que se obtuvieron desde el dron y las imágenes de objetos complementarias. Sin embargo, la cantidad no es suficiente para entrenar el algoritmo, y también es posible pasar las imágenes por el método de *Data Augmentation* con el que es posible realizar diferentes cambios a las imágenes como darles la vuelta en el eje vertical, hacerles zoom, variar el brillo, agregar ruido y también difuminar

la imagen (Ver Figura 37). Todo esto con el objetivo de que haya más variedad en la imagen y para que al momento de entrenar el algoritmo este reconozca las clases en diferentes situaciones o bajo ciertas condiciones. Este proceso se realizó con la herramienta Roboflow con la que es posible crear *dataset*s para algoritmos como YOLO, con tan solo agregar las imágenes y las etiquetas establecidas desde Cvat, para posteriormente dividir estas imágenes entre los siguientes sets:

- Entrenamiento: imágenes con las cuales se entrena el algoritmo.
- Validación: con estas imágenes se busca prevenir el *overfitting* y *underfitting*. En la detección de objetos se identifican los parámetros que dan mejor resultado para este grupo de imágenes.
- Pruebas: Imágenes con las cuales el modelo no ha sido entrenado, son "desconocidas" y permiten evaluar que tan eficazmente se está ejecutando.

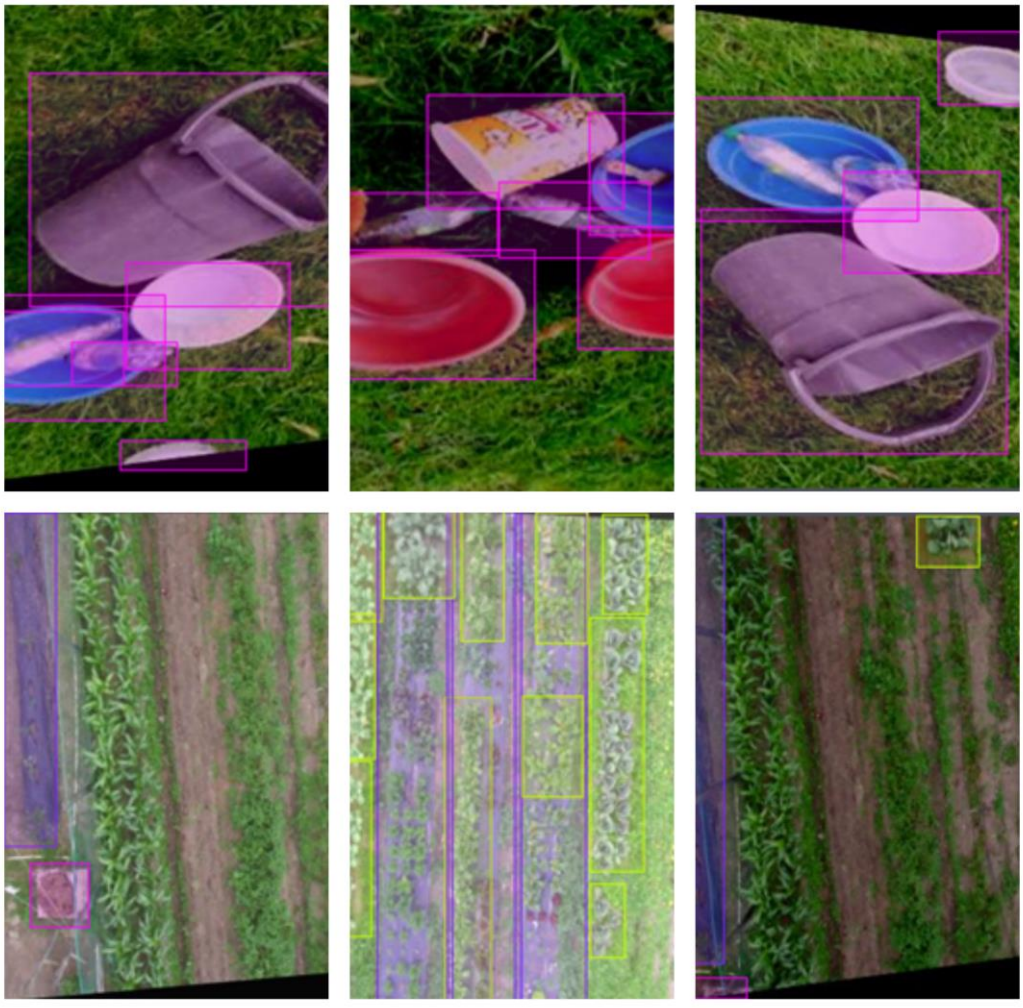


Figura 37. Data Augmentation aplicado en dataset.

9.2.3. Entrenamiento del algoritmo

Como ya fue mencionado, la versión de YOLO escogida fue la v4, más específicamente la correspondiente a Darknet. Esta red se entrenó con un documento en Google Colab disponible gracias a Roboflow, el cual contiene todas las configuraciones relacionadas a drivers de Nvidia, instalación de Darknet, configuración del *dataset*, entrenamiento y guardado de los pesos correspondientes a la red. Para este caso es posible ingresar en el código el link dado por Roboflow correspondiente al *dataset* previamente generado, o descargar el *dataset* para realizar algún cambio y cargarlo desde un servicio como Google Drive. El entrenamiento de la red tardó aproximadamente 6 horas, haciendo uso de una tarjeta gráfica con las características y drivers presentados en la Tabla 6.

Característica	Valor
Versión de NVIDIA-SMI	460.67
Versión de Driver Nvidia	460.32.03
Versión de CUDA	11.2
Tarjeta gráfica	Tesla T4

Tabla 6. Características de equipo y driver empleados para entrenamiento de YOLOv4.

Durante el entrenamiento se observa la precisión por cada 2000 iteraciones, el tiempo restante del entrenamiento y la mejor exactitud obtenida hasta el momento, que en este caso fue de 0.9410.

Al finalizar se indica cuál fue la precisión media y se guardan los pesos del entrenamiento. Es posible guardar los pesos con la mejor precisión o también los últimos obtenidos al terminar el entrenamiento. La precisión media alcanzada luego del entrenamiento fue de 0.8967.

9.2.4. Ejecución del algoritmo

Luego de haber obtenido los pesos del entrenamiento, la ejecución del algoritmo se hace desde otro programa en la máquina local, con el que se convierten los pesos obtenidos por el código de Darknet para poder utilizarlos en Tensorflow, también permite acceder a los datos

de las cajas delimitadoras a partir de las cuales se realiza el algoritmo de evasión de obstáculos. Con el programa manejado es posible emplear YOLOv4 en imágenes o video haciendo uso de Tensorflow.

Se realizaron pruebas con Tensorflow bajo diferentes *dataset*s, con lo cual se obtuvieron los resultados de las Tablas 7 y 8. Este *dataset* fue elaborado de forma que aquellos elementos que pudieran ser confundidos con objetos por parte del algoritmo, fueran etiquetados. De este modo, aquellas plantas y partes de cultivo, o secciones de bolsa que pudieran ser confundidos, serán reconocidos como tal, y se podrá diferenciar correctamente los elementos que si son objetos.

Versión de prueba de YOLOv4	Precisión media	Objetos detectados correctamente	Falsos positivos	Falsos negativos	No detectados	Precisión de detección de objetos
prueba_v1	0.68	498	8	45	104	0.76
prueba_v2	0.90	331	3	51	265	0.51
prueba_v3	0.90	483	12	0	40	0.91

Tabla 7. Resultados obtenidos al realizar pruebas de cada versión en distintas imágenes.

Versión de prueba de YOLOv4	No. Imágenes entrenamiento	No. Imágenes validación	No. Imágenes test
prueba_v1	492	44	11
prueba_v2	492	44	11
prueba_v3	1110	74	50

Tabla 8. Cantidad de imágenes de *dataset* utilizadas por prueba.

Para la Tabla 7, la diferencia entre cada versión de prueba es que en el *dataset* de cada una se etiquetó un número distinto de elementos correspondientes a plantas. Para la prueba_v1 la precisión media de la red fue de 0.68 aproximadamente, y en prueba_v3 fue de 0.9. Sin embargo, como se puede observar, para el caso de identificación de objetos, la red prueba_v1 tiene una mejor precisión, para ambos casos se trabajó con 492 imágenes de entrenamiento.

En el caso de la prueba_v3 se realizaron algunos ajustes al etiquetado, incluyendo la identificación de plantas que estaban sobre las bolsas y de cualquier barra de madera que se encontrara en el cultivo, pero sin clasificar todas las plantas que se pudieran encontrar en las imágenes. Sin embargo, el mayor cambio fue que se trabajó con un *dataset* de 1110 imágenes, mucho mayor a las pruebas previas, en la Tabla 8 se puede ver más a detalle estos datos.

9.2.5. Detección realizada por el algoritmo

Para hacer uso del algoritmo solo se requiere ejecutar la función de detección que tiene como parámetro de entrada una imagen. Esta función realiza el proceso de detección, calcula las cajas delimitadoras y devuelve una imagen en la que los objetos encontrados son encerrados en una caja junto con el valor de precisión de la detección, resultado que se presenta en la Figura 38, en donde se evidencia el resultado de este algoritmo para la versión de prueba 3 aplicado a algunas imágenes con distinta resolución.

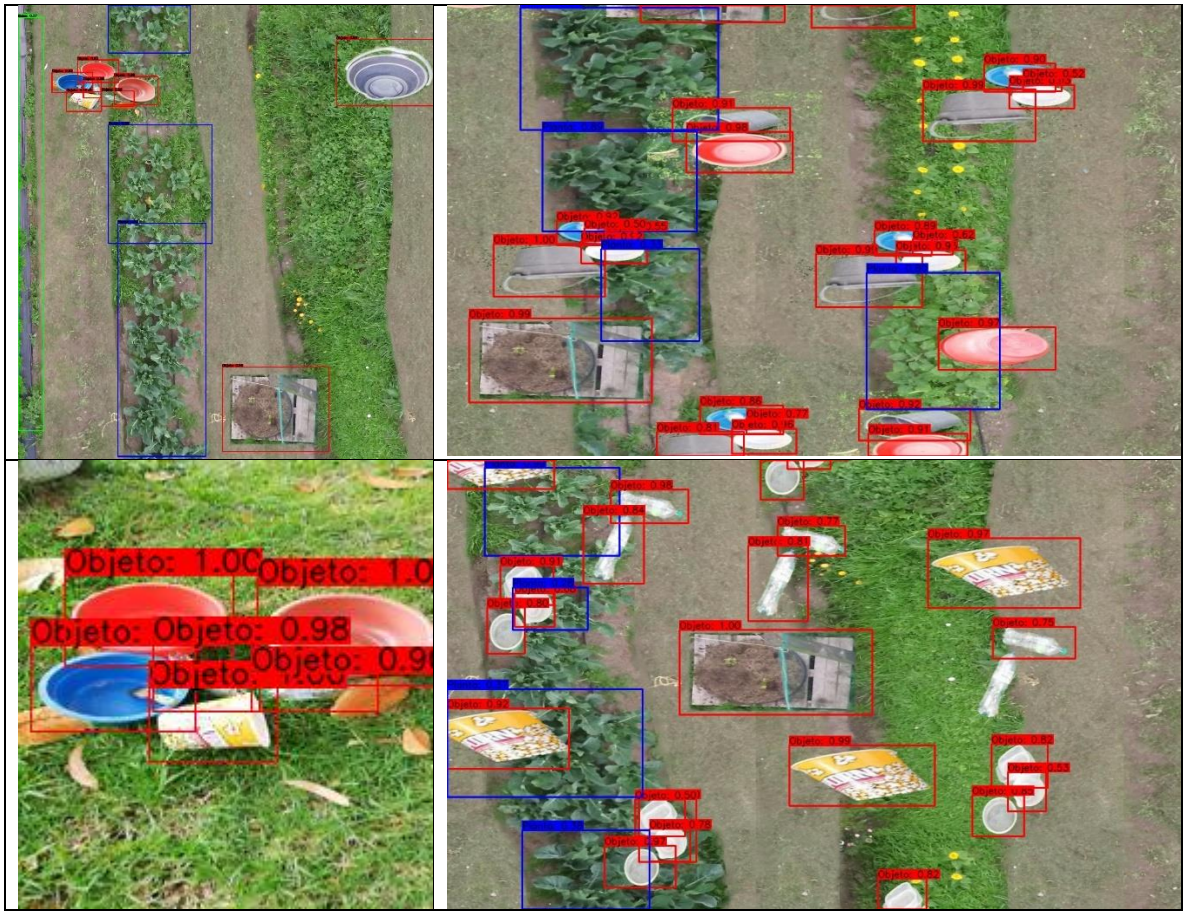


Figura 38. Detección realizada por algoritmo YOLOv4 en imágenes de prueba.

10. Evasión de obstáculos

Cuando se obtiene la posición de los objetos mediante el algoritmo YOLOv4, es posible identificar cuáles de estos obstaculizan el recorrido que se establece mediante los centros del cultivo, para de esta forma generar una nueva trayectoria que permita evadirlos, esto se realizó haciendo uso del algoritmo A*. Para evitar colisiones se debe tener en cuenta tanto el tamaño del dron como el de las cajas delimitadoras de los objetos, para simplificar el problema de manejar estos dos tamaños se aumentó el espacio que ocupan los objetos en la imagen en función del que ocupa el dron, y a esto también se le realizó un aumento adicional por seguridad. De esta forma solo se requiere calcular un camino de un punto del cultivo a otro entre objetos "más grandes".

*10.1. Algoritmo A**

Este algoritmo es usado para búsqueda de caminos, permite encontrar el camino más corto y hace uso de una heurística para guiarse.

Implementa una función de cálculo de costo descrita de la siguiente forma:

$$f(n) = g(n) + h(n)$$

En donde $g(n)$ representa el costo exacto desde el punto inicial hasta alguno de los vértices o nodos n, y $h(n)$ representa el costo heurístico estimado desde el nodo n hasta el punto final o meta. http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

El proceso realizado para este proyecto consiste en el cálculo de valores locales y globales desde cada uno de los nodos, y la asignación de un nodo padre para los nodos que a partir de las conexiones entre ellos forman un camino o trayectoria, este cálculo se realiza a partir de valores de distancias entre nodos y usando heurística, como lo describe la función de cálculo de costo. Los valores locales y globales se van actualizando a partir del análisis de cada conexión entre los nodos existentes; cuando todas las conexiones de cada nodo son analizadas (en orden, siempre tomando como siguiente el nodo con menor valor global) se obtiene el camino más corto entre el nodo inicial y el nodo objetivo.

Dentro de los nodos para tener en cuenta se encuentran los vértices de cada caja delimitadora detectada, esto quiere decir que pueden ser parte de la trayectoria final con la que se evaden los objetos si el algoritmo identifica que al pasar por estos puntos la trayectoria que se genera es óptima.

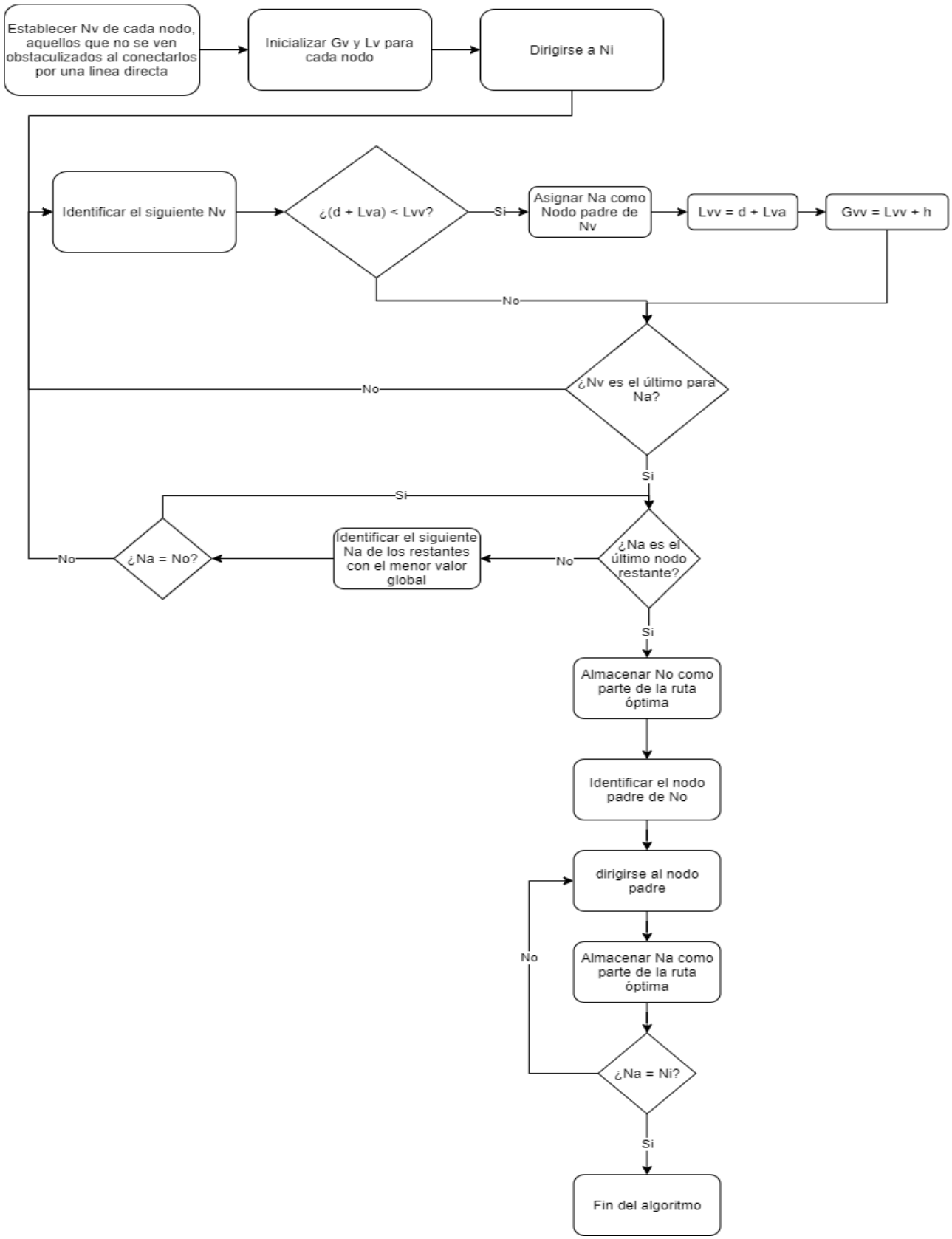


Figura 39. Diagrama que describe el algoritmo A.*

En el diagrama de la Figura 39 se explica cómo fue implementado el algoritmo. Para este caso, se establecen las definiciones de la Tabla 9:

Abreviación	Descripción
Na	Nodo actual
Nv	Nodo vecino
Lva	valor local del nodo actual
Lvv	Valor local del nodo vecino
Gvv	Valor global del nodo vecino
Ni	Nodo inicial
No	Nodo objetivo
d	distancia de Na a Nv
h	valor heurístico que se define como la distancia directa entre el nodo vecino y el nodo objetivo.

Tabla 9. Definiciones para diagrama de algoritmo A*

A* fue utilizado para recalcular una ruta en la que hay presencia de objetos, buscando mantener el recorrido planeado previamente. La forma en la que se aplicó el algoritmo se ve evidenciada en el diagrama de la Figura 40.

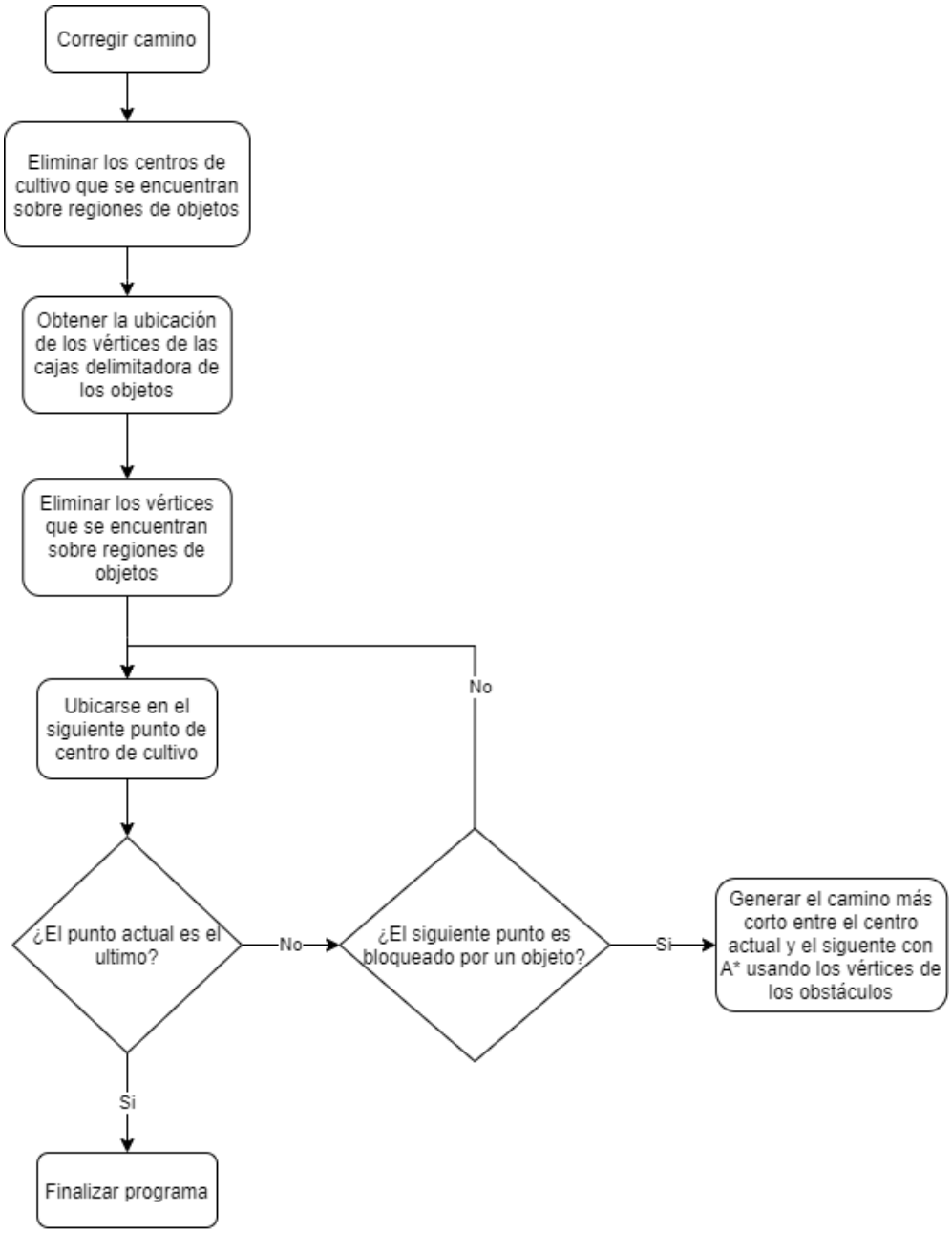
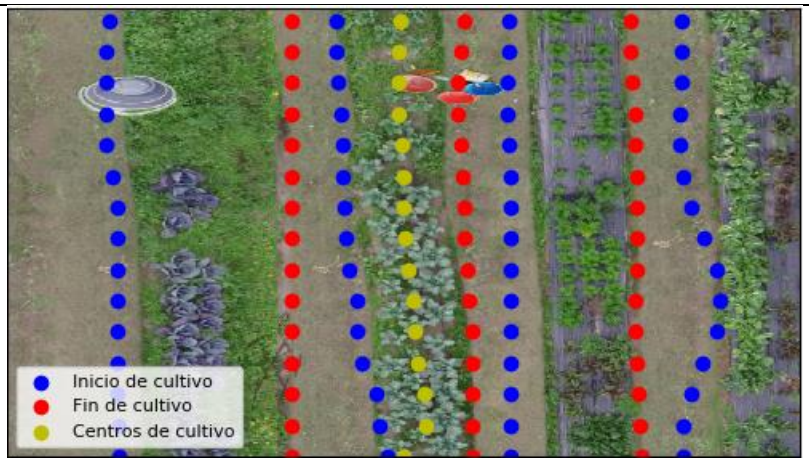
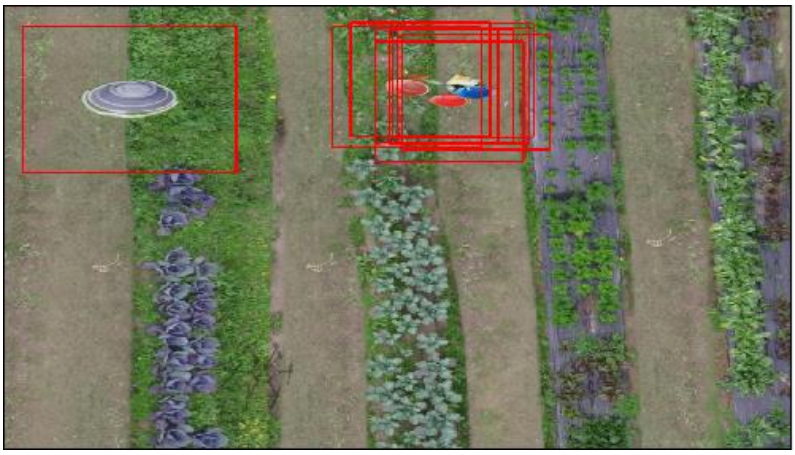


Figura 40. Diagrama del proceso de evasión de obstáculos en el que se implementa A para corregir el camino trazado para el dron.*

Luego de aplicar el algoritmo de evasión de objetos, la trayectoria obtenida en un principio es modificada, este cambio se evidencia en la Figura 41.



(a) Elementos de cultivo y trayectoria calculada



(b) Objectos detectados con cajas delimitadoras calculadas



(c) Trayectoria para el dron ajustada

Figura 41. Ajuste de trayectoria para el dron a partir de evasión de obstáculos.

Como se puede observar, los vértices de las cajas delimitadoras del cúmulo de objetos encontrados en el cultivo se convirtieron en parte del camino que debe recorrer el dron para

desviar los obstáculos, esto debido a que fueron considerados por el algoritmo como los puntos óptimos para generar una nueva trayectoria. Es importante tener en cuenta que el tamaño de la caja delimitadora fue aumentado para tener una seguridad extra y puede ser modificada dentro del programa.

11. Integración del simulador con el proceso de detección de cultivos

Dentro de esta sección se expone la consolidación final del proceso de toma de imágenes, detección de cultivo, detección de objetos y desplazamiento del dron dentro del ambiente simulado de Gazebo. Se dan a conocer el proceso, las utilidades para relacionar las imágenes y el simulador, los problemas que se presentaron y las soluciones que se dieron al momento de integrar el simulador con los resultados de procesamiento de imagen y llevar a cabo con éxito el desplazamiento del dron hacia las ubicaciones del cultivo deseadas.

11.1. Uso de API REST para implementar algoritmo YOLOv4 en el simulador

La simulación en ROS Kinetic hace uso de Python2, una versión un poco vieja teniendo en cuenta que para YOLOv4 se hace uso de Python3. Esto genera algunos conflictos debido a que hay funciones que se tratan de usar en el algoritmo, pero no están en la versión anterior. Realizar una actualización a una versión de ROS más reciente no era opción debido a que en el repositorio de RotorS no se presenta otra opción además de ROS Kinetic y ROS indigo (una versión más vieja) para usar este simulador. Por esta razón se presentó como solución crear un API REST que está activa en una máquina externa que cumpla con la versión de Python, que además puede tener características de hardware mejores que el equipo desde el cual se ejecuta la simulación, como lo fue en este caso.

Las API son un conjunto de definiciones y protocolos que permiten interactuar con un equipo para ejecutar una función [64]. En este caso, se utilizó una API REST simple en la que solamente se envió información mediante un formato JSON hacia una máquina que ejecuta el proceso de detección de objetos mediante YOLOv4.

Esta API tiene un servicio que recibe una imagen codificada, la decodifica para posteriormente realizar todo el proceso de detección de objetos y devuelve la información de cajas delimitadoras requerida para que, desde la máquina que está realizando la simulación, se puedan establecer las coordenadas de los objetos detectados (ver Figura 42). Este proceso permitirá ejecutar el algoritmo de evasión de objetos y también mostrar en la imagen del cultivo cuál es el área detectada para cada objeto.

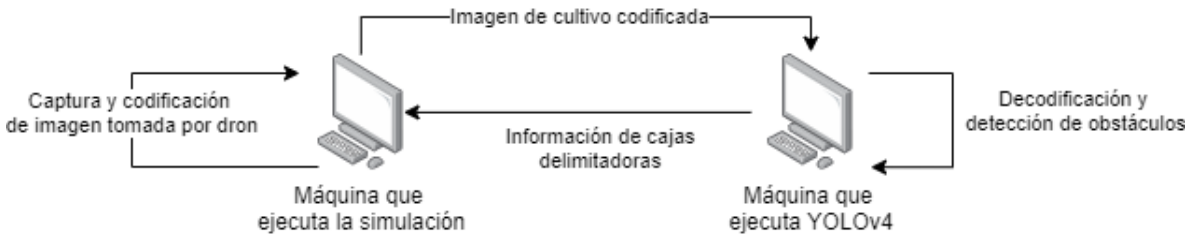


Figura 42. Comunicación de equipo de simulación y equipo con algoritmo YOLOv4 para obtener coordenadas de objetos detectados.

11.2. Equivalencia Píxel – Distancia

Dado que los caminos de cultivo y los objetos son detectados en una imagen de 1080x720, se requiere convertir las distancias y ubicaciones representadas en píxeles, a unidades que pueda interpretar el simulador para desplazar el dron, que para este caso son metros.

Para el cálculo de la equivalencia píxel-distancia se usaron imágenes tomadas por el dron en el simulador a distintas alturas, la imagen posee líneas en el suelo que se encuentran divididas a 1m de acuerdo con las configuraciones generales de Gazebo.

Posteriormente a partir del visualizador de imagen de Matplotlib se establecieron las distancias en píxeles entre dos líneas en el simulador para cada una de las alturas. Se generó una tabla de datos y mediante WolframAlpha se realizó una regresión que evidenció un comportamiento no lineal, el cual, para una altura entre 1 y 5.5 metros se puede describir con la siguiente ecuación con x como la altura a la que se encuentra el dron:

$$Equivalencia = 0.0268078x^9 - 0.755556x^8 + 9.10688x^7 - 61x^6 + 245.48x^5 - 588.758x^4 + 728.495x^3 - 72.9861x^2 - 1009.61x + 1174$$

Esta equivalencia se usa para determinar cuántos píxeles hay en 1 metro a determinada altura. A partir de esto ya es posible determinar la distancia a la que se debe desplazar el dron cuando se quiera que este llegue a cada uno de los centros detectados en las imágenes tomadas, después de realizar todo el proceso de detección de texturas, bordes y centros. Para esto se debe tener en cuenta que la ubicación del dron siempre se mantiene en el centro de la imagen, dada la posición en la que se ubicó la cámara agregada al simulador.

11.3. Desplazamiento del dron en simulación

Por cada fotograma que captura el dron se realiza el proceso de detección de cultivos para posteriormente encontrar los límites y centros del cultivo. Estos centros son detectados solamente para el cultivo más cercano al que se encuentra el dron. El camino obtenido se ajusta luego de pasar por el algoritmo de evasión de obstáculos y queda listo para ser usado en el proceso de desplazamiento del dron.

Para desplazar el dron se requieren los datos de posición actual en X y Y del dron y la posición en X y Y de cada uno de los centros detectados dentro de la imagen. Dado que la cámara se encuentra en el centro del dron, solamente se tomarán en cuenta para el desplazamiento aquellos centros del cultivo que se encuentren por encima de la mitad de la imagen.

A partir de los datos de posiciones, se hace uso de una función que emplea la equivalencia Píxel – Distancia para identificar las posiciones reales de los centros de cultivo. Luego de esto, en un ciclo se envía por iteración la posición de cada uno de los centros, mientras que el dron no haya llegado a la posición deseada no se hará nada; una vez llegue a uno de los centros se continúa con el otro hasta que finalmente se llegue al último. Terminado este proceso se vuelve a tomar la imagen de la cámara del dron y se ejecuta nuevamente la detección del cultivo y el desplazamiento. Este proceso se describe en el diagrama de la Figura 43.

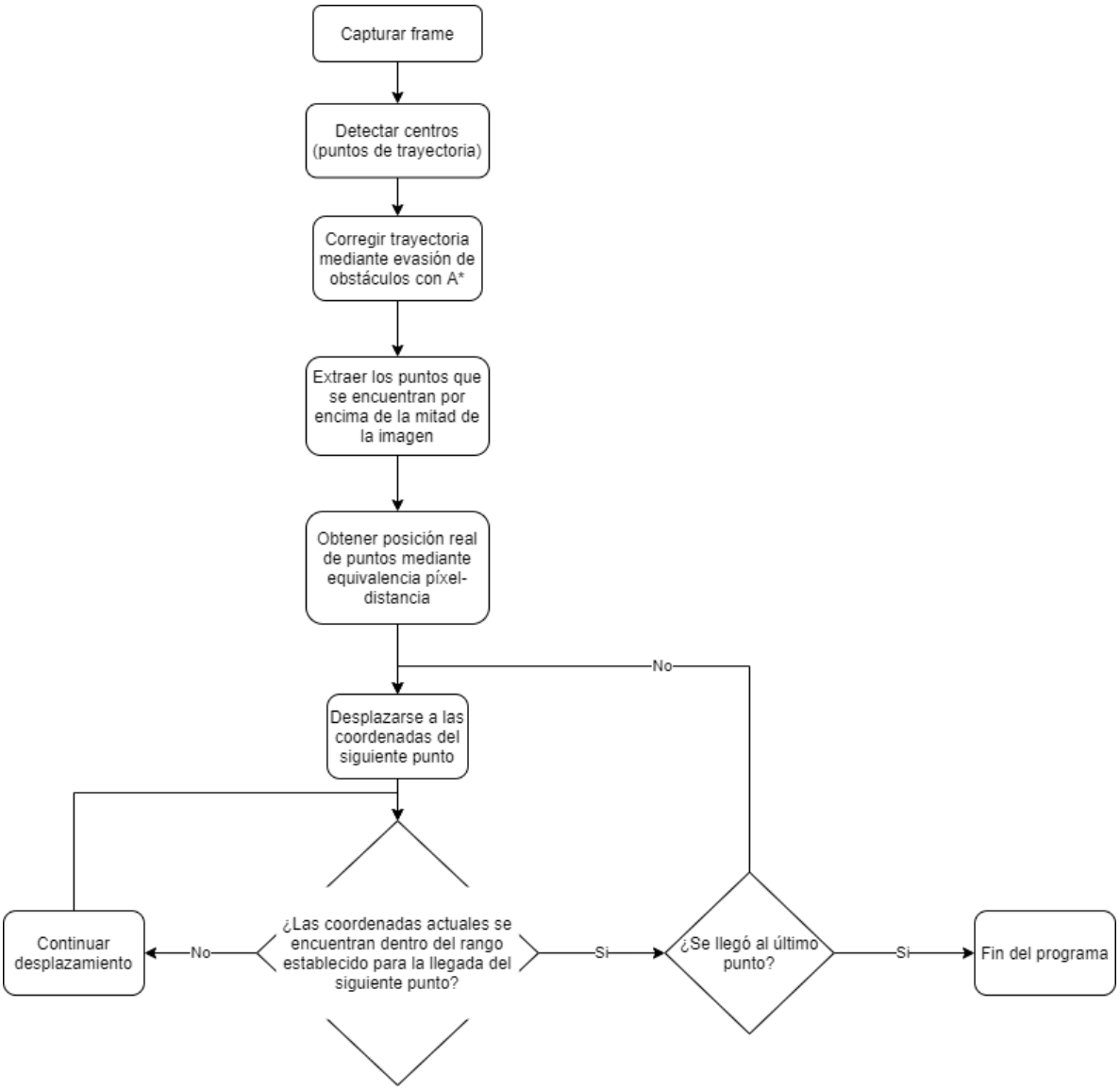


Figura 43. Diagrama del proceso de evasión de obstáculos en simulación en Gazebo.

12. Generación de mapa

Uno de los objetivos de este proyecto es generar un mapa a partir del recorrido que realice el dron donde puedan ser identificados los elementos del cultivo.

Para realizar esto se almacenaron en un archivo txt los datos obtenidos en cada fotograma en el que se detectaron los límites, camino de cultivo y objetos. Estos datos pasaron por un proceso de conversión a posición real mediante la equivalencia píxel-distancia empleada previamente, por lo que las distancias se encuentran en metros, respecto al origen de la simulación.

Teniendo generado el archivo, se desarrolló un script que permitiera graficar los elementos almacenados, esto se visualiza en la Figura 44 en la que se graficó una parte del archivo generado y se añadió la imagen del cultivo correspondiente a la sección graficada.

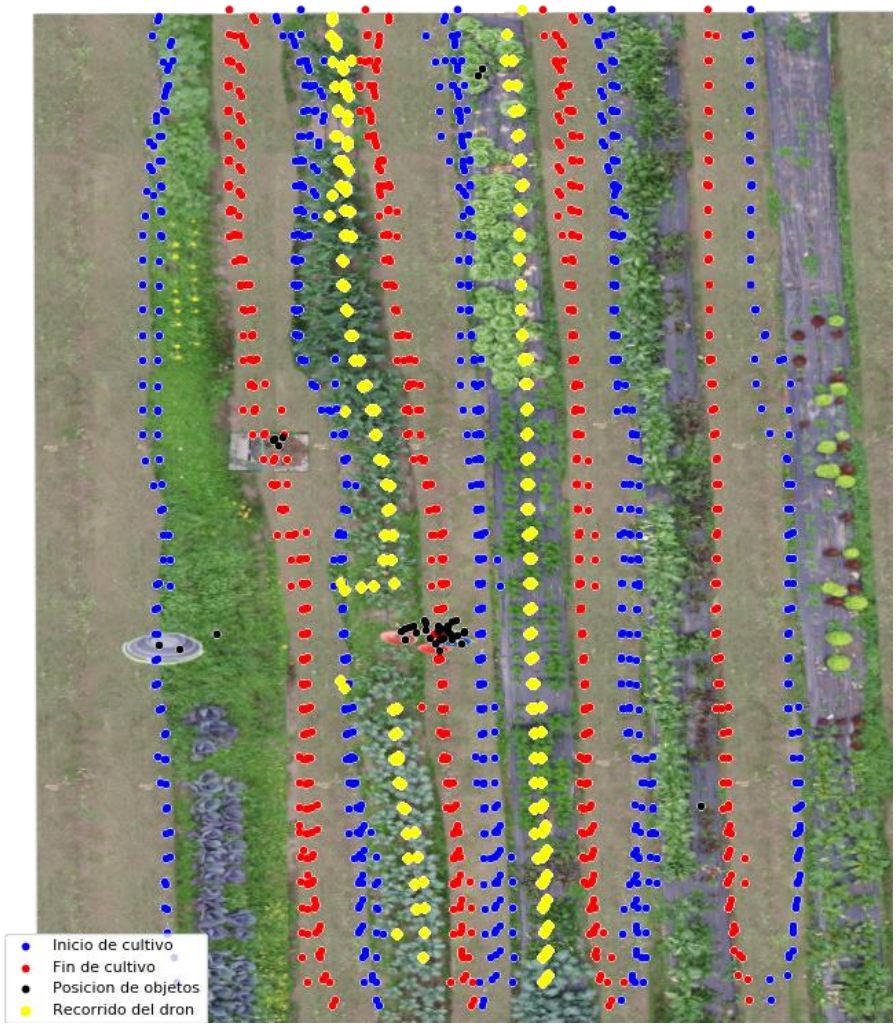


Figura 44. Mapa de elementos en cultivo a partir de datos de simulación.

13. Resultados

Luego de la implementación de la simulación en el ambiente de Gazebo, se tomaron datos del recorrido del dron, para compararlos contra la señal de control enviada. Los datos tomados fueron la posición en X, Y, Z deseada, y la posición en la que se movió el dron. Las gráficas obtenidas son resultado de calcular la distancia entre el punto inicial en el que se encontraba el dron y cada una de las posiciones en los tres ejes.

En la Figura 45 se observa cómo el dron se desplaza desde unas condiciones iniciales cualquiera, en este caso su posición difería de la posición deseada en los tres ejes, para el eje Z el dron se encontraba aproximadamente 1 metro por encima de la altura con la que se realiza todo el proceso de detección de cultivo.

Luego de estos resultados, en el caso en que el dron se encuentre en condiciones iniciales, se consideró conveniente realizar un primer desplazamiento en el eje Z hasta la altura deseada y luego desplazarse hasta los puntos deseados en el plano X y Y; esto debido a que, en caso de que un objeto se encuentre cerca a los primeros puntos de desplazamiento, podrían generarse resultados no esperados como choques con el objeto dada la lógica con la que se tiene programado el proceso de detección de objetos, conversión de distancias y desplazamiento del dron. Teniendo en cuenta esto, se presentan las gráficas de la Figura 45 y Figura 49 de forma demostrativa para visualizar los desplazamientos y velocidades de motores en caso de un desplazamiento que involucre moverse en 3 dimensiones.

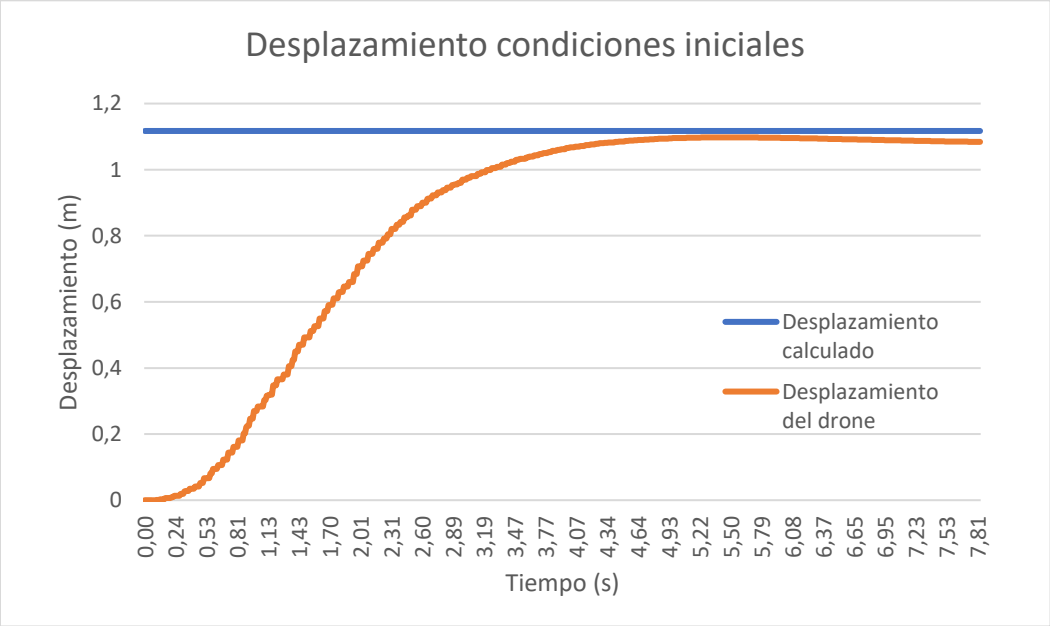


Figura 45. Desplazamiento del dron desde condiciones iniciales.

En la Figura 46 se presenta la gráfica del desplazamiento del dron hacia el primer y segundo centro de cultivo detectado, respectivamente.

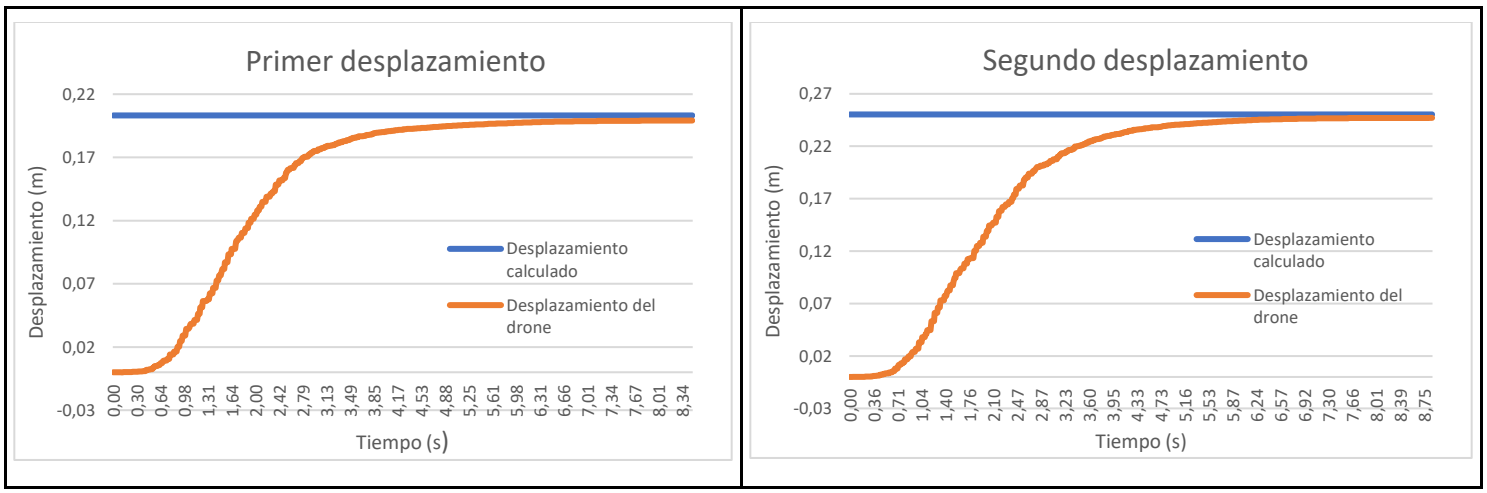


Figura 46. primer y segundo desplazamiento del dron.

También se buscó evidenciar el escenario en el que el movimiento del dron se viera afectado por alguna perturbación, de manera que mediante código en una parte del recorrido mientras el dron se desplaza, se realizó un cambio de la velocidad en sus motores durante un 1 segundo, para de esta forma verificar cómo reacciona y qué comportamiento tiene para llegar al centro de cultivo que le corresponde. Esto se evidencia en la Figura 47.

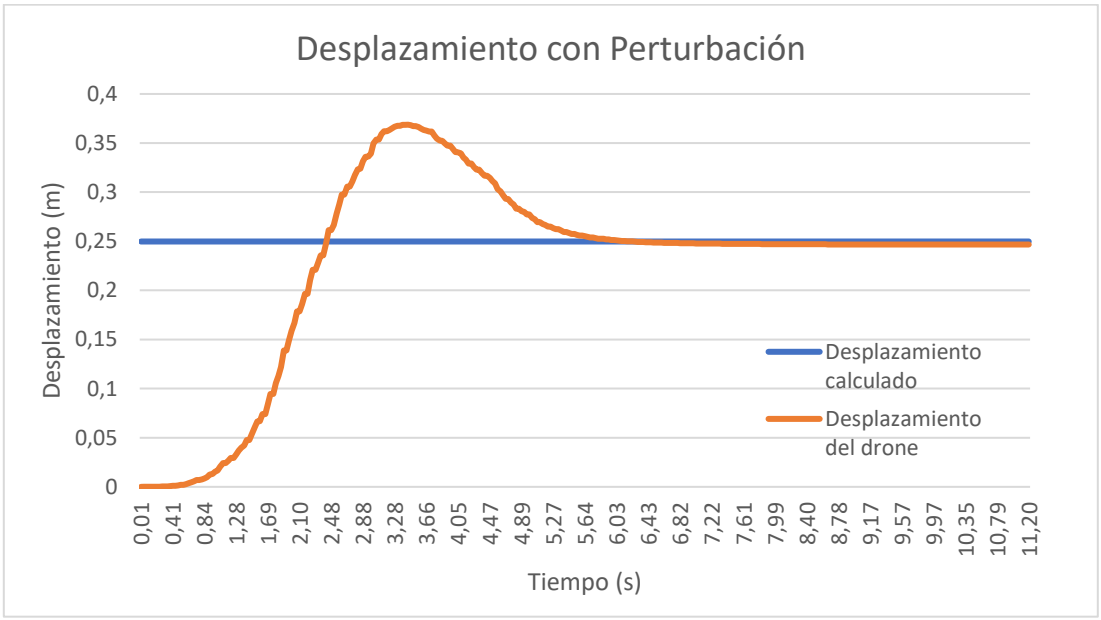


Figura 47. Desplazamiento del dron en el que se presenta una perturbación.

Para cada uno de estos desplazamientos se realizó su respectiva gráfica de error, presentada en la Figura 48.

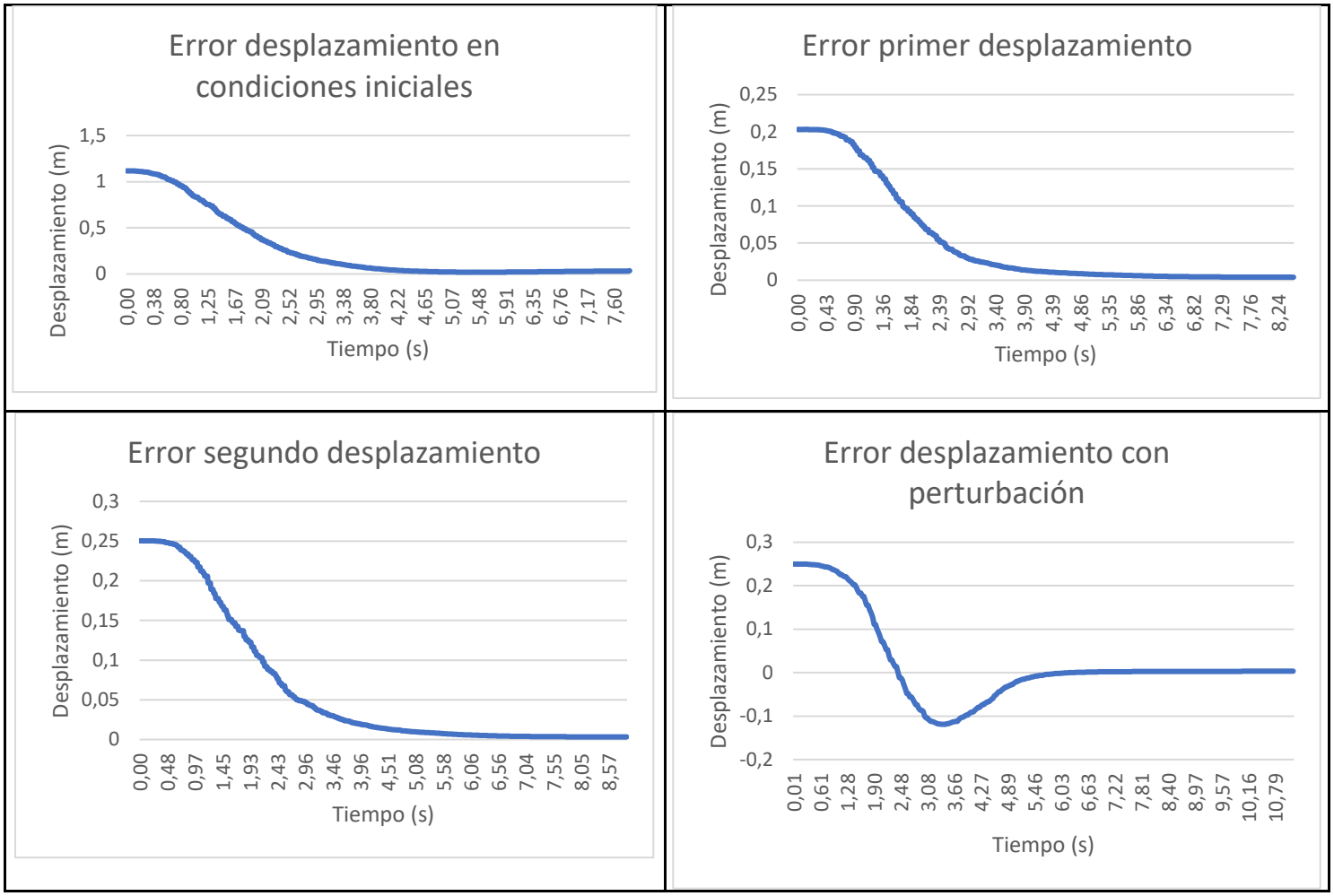


Figura 48. Gráfica de error para cada desplazamiento.

Para recorrer los centros del cultivo, el dron se desplaza hacia cada uno de los centros de forma discontinua, esto quiere decir que al llegar a uno de los centros realiza una pausa antes de continuar hacia el otro centro. Esto sucede porque el dato de desplazamiento que se le envía es una lista de posiciones a las que debe llegar, por lo que hasta que no llegue a una de las posiciones en la lista no comienza a desplazarse a la siguiente. Se tomaron datos de las velocidades de cada motor para examinar si se está realizando un cambio brusco de velocidad en la pausa y continuación del desplazamiento (ver Figuras 49 y 50), lo que podría evidenciar un consumo alto de corriente que a su vez afectaría a los motores y la autonomía del dron durante el proceso de identificación de surcos.

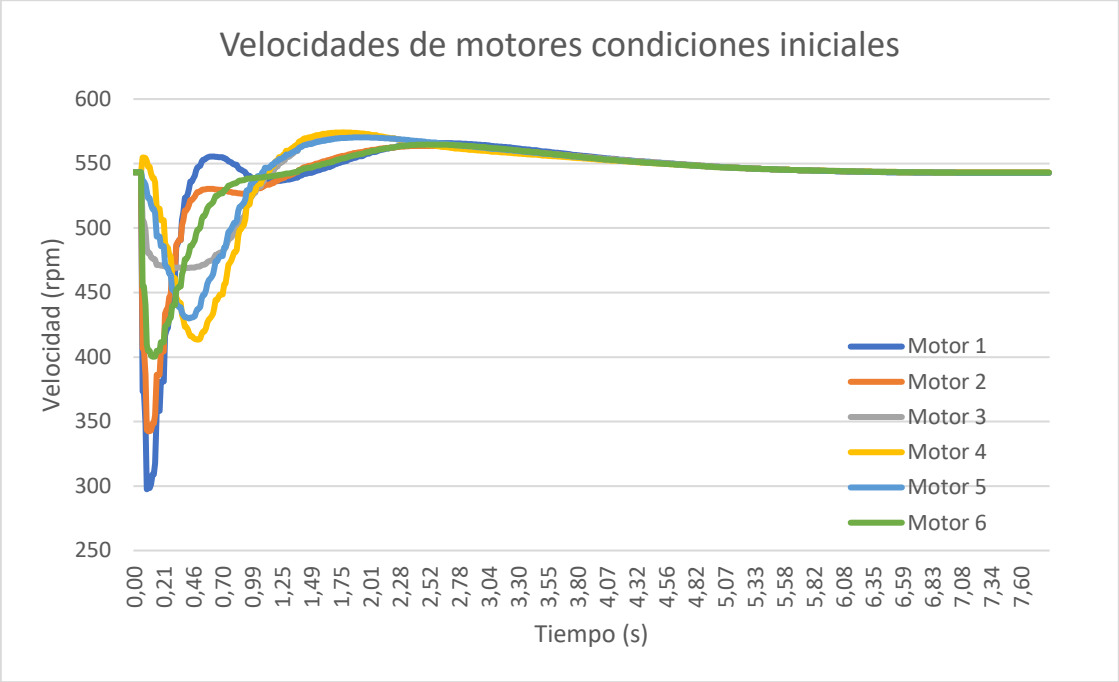


Figura 49. Velocidades de los motores del dron al desplazarse desde condiciones iniciales.

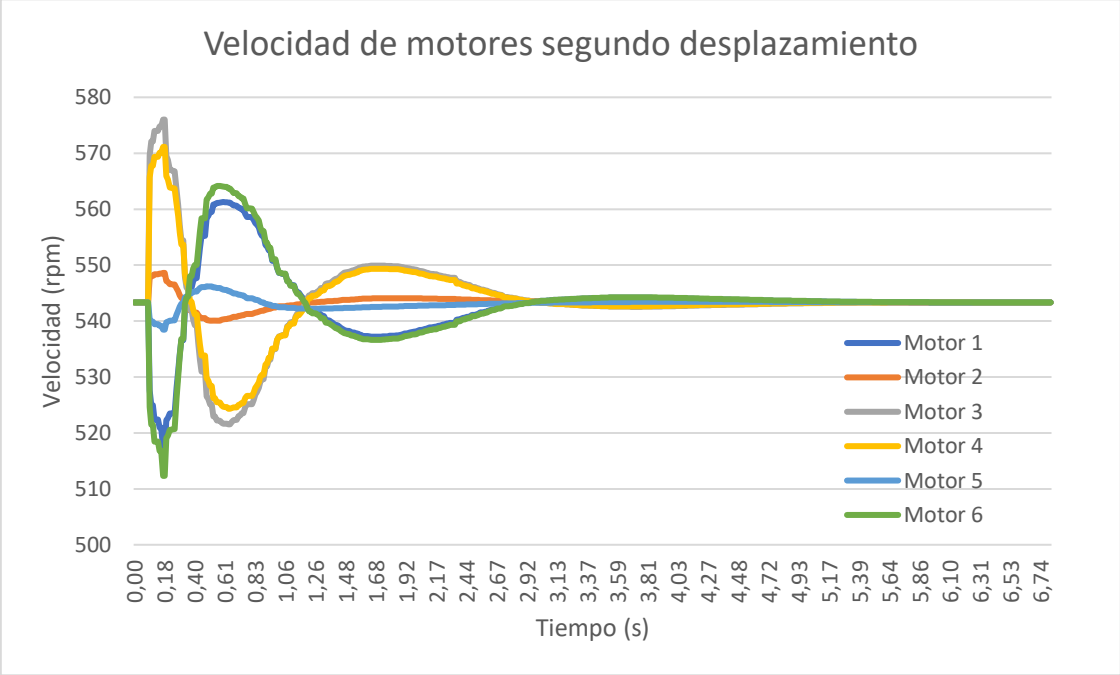


Figura 50. Velocidades de los motores del dron al desplazarse hacia el segundo punto.

En la Figura 49 se observa que el máximo cambio de velocidad se presenta cuando uno de los motores alcanza 300 rpm, lo cual indica que un motor bajó aproximadamente de 540 a 300 rpm en 0.1 segundos. Esto para el caso en el que había unas condiciones iniciales del dron en las que este se encontraba en una altura superior a la parametrizada para realizar el recorrido.

Por otra parte, en la Figura 50 se observa que uno de los motores bajó de aproximadamente 540 rpm a 512 rpm en 0.18 segundos, esta fue el máximo cambio en la velocidad alcanzado en un desplazamiento de un centro a otro en el que el dron ya se encontraba a la altura parametrizada.

14. Conclusiones y trabajos futuros

El procesamiento de imagen realizado a los datos recibidos por la cámara del dron requirió de un estudio previo de la literatura y un ensayo de cada uno de los filtros y operaciones aplicadas a la imagen. Se verificó cuáles parámetros (de cada una de las operaciones) eran los más aptos para obtener los cambios en la imagen, que al relacionarlos entre sí permitiera delimitar adecuadamente los elementos del cultivo. El resultado fue una imagen binaria en la que se logró diferenciar los elementos verdes del cultivo, construyendo trazos que representan las líneas de cultivo dentro de una imagen binaria.

Para el caso de las propiedades de la matriz de co-ocurrencia, se observó que los resultados no fueron lo suficientemente satisfactorios para emplear la red neuronal. Esto puede ser debido a la calidad de las imágenes de texturas con las que fue entrenada, los hiperparámetros que fueron seleccionados, la arquitectura escogida, o que esto en conjunto a las características de la matriz de co-ocurrencia seleccionadas no funciona adecuadamente para este problema en particular.

Por otro lado, trabajar directamente con el contenido de la imagen empleando una red neuronal convolucional permitió diferenciar correctamente aquellos elementos que fueran cultivo de los que no, estableciendo las bases para delimitar los surcos y construir un recorrido para el dron. Si bien se observó que la precisión de la red no está por encima del 0.9, esto es debido a la detección incorrecta de parte de los objetos incluidos dentro del *dataset*. Aun así, la gran mayoría de elementos pertenecientes a cultivo, sean “plantas”, “pasto” o “bolsa” fueron detectadas correctamente y diferenciadas del suelo; se cumplió con el principal propósito de esta red, el cual fue identificar plantas y otras texturas como las bolsas sobre las que crece parte del cultivo, con lo cual se complementaron aquellos elementos faltantes en la imagen binaria de la primera etapa de procesamiento de imagen.

Se ideó un algoritmo para identificar los inicios y finales de cada cultivo, así como el punto medio de estos, que corresponde al camino por el que se desplaza el dron. Para que este algoritmo funcione correctamente se debe tener un apropiado procesamiento de imagen en el que se encuentren bien definidas las líneas que delimitan al cultivo, pertenecientes a la imagen binaria.

Se requirió de un algoritmo adicional para la detección de obstáculos ya que, con la red convolucional, si bien también puede ser usada para esto, se observó que para este caso de uso no se obtuvieron resultados satisfactorios con la clase “objeto”. Por esta razón se hizo uso de una inteligencia artificial especializada en esta clase de problemas, empleando YOLOv4 se obtuvo la posición de los objetos y una medida aproximada del espacio que ocupan dentro de la imagen que toma el dron desde el aire. Los resultados del entrenamiento del algoritmo después de algunas pruebas fueron suficientes para detectar correctamente diferentes objetos que se podrían encontrar en un cultivo.

Una vez se obtuvo el camino que el dron debe recorrer y la posición de los obstáculos se realizó la etapa de evasión de obstáculos, en la que empleó el algoritmo de búsqueda A* para buscar la mejor ruta entre cada par de puntos separados por un obstáculo; durante esta etapa el camino del dron es corregido de forma que se pueda evadir los obstáculos, por lo tanto, hay puntos de la ruta previa que son eliminados y otros nuevos pertenecientes a los vértices de las cajas delimitadoras de los objetos. En caso de que no se encuentre una ruta para evadir el obstáculo, el recorrido será cortado hasta el punto previo a la evasión del obstáculo. Este es un caso poco probable teniendo en cuenta que para que esto suceda tendría que haber una gran cantidad de objetos en medio del cultivo o distintos objetos de gran tamaño ubicados de forma que no le dieran espacio al algoritmo para generar una trayectoria.

Durante los recorridos de la simulación los objetos encontrados dentro del cultivo fueron evadidos exitosamente, el dron avanzó por los caminos de cultivo sin problema.

Para trabajos futuros se propone explorar alternativas a la red convolucional usada para la identificación de bolsas y de plantas empleada en este proyecto. Aunque la técnica de matriz de co-ocurrencia utilizada no funcionó adecuadamente, puede que otros algoritmos o herramientas de segmentación de textura funcionen para este caso de uso y sería útil evaluar los resultados que estos conlleven. Además de esto, también se sugiere evaluar la posibilidad de implementar versiones más recientes del algoritmo YOLO como, por ejemplo, YOLOv5 o versiones futuras que en algún momento se lleguen a crear. Adicionalmente, es posible enriquecer la clase objeto en el entrenamiento del algoritmo YOLO con más imágenes en las que haya elementos que se puedan considerar obstáculos.

En lo que respecta a este proyecto, los algoritmos y la simulación servirán como base para una plataforma física, que le dará una perspectiva aérea sobre el terreno que tendrá que recorrer la plataforma Ceres_AgroBot, proporcionando información de líneas de cultivo, posición de objetos y la trayectoria seguida por el dron durante la exploración del cultivo.

15. Bibliografía

[1] "El 17% del total de la fuerza laboral del país trabaja en el campo", Portafolio.co, 2021. [Online]. Disponible: https://www.portafolio.co/economia/el-panorama-de-la-agricultura-en-colombia-en-su-dia-internacional-de-la-agricultura-544437. [Accedido 10 Julio 2021].

[2] T. Moribe, H. Okada, K. Kobayashl and M. Katayama, "Combination of a wireless sensor network and dron using infrared thermometers for smart agriculture," 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2018, pp. 1-2.

[3] D. Murugan, A. Garg and D. Singh, "Development of an Adaptive Approach for Precision Agriculture Monitoring with Dron and Satellite Data," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 10, no. 12, pp. 5322-5328, Dic. 2017.

[4] C. Cadena et al., "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," in IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309-1332, Dic. 2016.

[5] J. Park, S. Im, K. Lee and J. Lee, "Vision-Based SLAM System for Small UAVs in GPS-Denied Environments", Journal of Aerospace Engineering, vol. 25, no. 4, pp. 519-529, 2012. Disponible: 10.1061/(asce)as.1943-5525.0000160 [Accedido 20 enero 2020].

[6] T. Zuo, H. Min, Q. Tang and Q. Tao, "A Robot SLAM Improved by Quantum-Behaved Particles Swarm Optimization", Mathematical Problems in Engineering, vol. 2018, pp. 1-11, 2018. Disponible: 10.1155/2018/1596080 [Accedido 20 enero 2020].

[7] D. Murugan, A. Garg, T. Ahmed and D. Singh, "Fusion of dron and satellite data for precision agriculture monitoring", 2016 11th International Conference on Industrial and Information Systems (ICIIS), 2016. Disponible: 10.1109/iciinfs.2016.8263068 [Accedido 30 julio 2019].

[8] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi and C. Avizzano, "Towards Smart Farming and Sustainable Agriculture with Drones", 2015 International Conference on Intelligent Environments, 2015. Disponible: 10.1109/ie.2015.29 [Accedido 30 julio 2019].

[9] J. Stafford, "Implementing Precision Agriculture in the 21st Century", Journal of Agricultural Engineering Research, vol. 76, no. 3, pp. 267-275, 2000. Disponible: 10.1006/jaer.2000.0577 [Accedido 25 julio 2019].

[10] D. Avendaño, A. Molina, C. Pulido y N. Velasco, “Control y estabilización de altura de un Quadrotor por Modos Deslizantes”, en IV Congreso internacional de Ingeniería Mecatrónica y Automatización, 2015, pp. 2074-282.

[11] M. Molina, D. Avendaño, C. Pulido y L. Solaque, “Sistema de Transporte Cooperativo Desarrollado Para un Grupo de Robots Móviles No-Holonómicos Usando el Método Líder Virtual”, en IV Congreso internacional de Ingeniería Mecatrónica y Automatización, 2015, pp. 264-272.

[12] D. Sogamoso, G. Solaque, y J. Aponte, “Integración de la mecatrónica al desarrollo de la agricultura de precisión aplicada al control mecánico de maleza”, en IV Congreso internacional de Ingeniería Mecatrónica y Automatización, 2015, pp. 302-312.

[13] C. Pulido-Rojas, M. Molina-Villa and L. Solaque-Guzmán, "Machine vision system for weed detection using image filtering in vegetables crops", Revista Facultad de Ingeniería Universidad de Antioquia, no. 80, 2016. Disponible: 10.17533/udea.redin.n80a13 [Accedido 15 mayo 2020].

[14] Dron para toma de imágenes aéreas Matrice 600. Aeroexpo.

[15] D. Yallappa, M. Veerangouda, D. Maski, V. Palled and M. Bheemanna, "Development and evaluation of dron mounted sprayer for pesticide applications to crops", 2017 IEEE Global Humanitarian Technology Conference (GHTC), 2017. Disponible: 10.1109/ghtc.2017.8239330 [Accedido 15 agosto 2019].

[16] Tecnovant.co. “Quienes Somos? – Tecnovant”. [Online]. Disponible: https://www.tecnovant.co/index.php/quienes-somos/.

[17] PwC. (2017). “Soluciones basadas en drones para la agricultura”. [Online]. Disponible: https://www.pwc.com/co/es/assets/document/Soluciones%20basada%20en%20drones%20para%20agricultura%20VF1.pdf

[18] J. Berrío, B. Posso, J. Perafan y E. Caicedo, “Mapeo y localización simultánea de un robot móvil en ambientes estructurados basado en integración sensorial”. MEng. tesis, Universidad del Valle, Cali, Valle del Cauca, 2015.

[19] C. Velásquez, J. Chávez, y E. Córdoba. “Implementación de sistemas de navegación autónomo en robot móvil experimental para reconstrucción y exploración de entornos desconocidos”. en IV Congreso Internacional De Ingeniería Mecatrónica Y Automatización -CIIMA 2015, 2015, pp. 73-74.

[20] J. Berrío, E. Caicedo, y L. Paz, “Mapeo y localización simultáneo de un robot para levantamiento de mapas en interiores”, en Eleventh LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI’2013), Cancún, México, 2018.

[21] R. Mardiyanto, J. Anggoro and F. Budiman, "2D map creator for robot navigation by utilizing Kinect and rotary encoder", 2015 International Seminar on Intelligent Technology and Its Applications (ISITIA), 2015. Disponible: 10.1109/isitia.2015.7219957 [Accedido 31 julio 2019].

[22] N. Habibie, A. Nugraha, A. Anshori, M. Ma'sum and W. Jatmiko, "Fruit mapping mobile robot on simulated agricultural area in Gazebo simulator using simultaneous localization and mapping (SLAM)", 2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS), 2017. Disponible: 10.1109/mhs.2017.8305235 [Accedido 18 agosto 2019].

[23] J. Moonrinta, S. Chaivivatrakul, M. Dailey and M. Ekpanyapong, "Fruit detection, tracking, and 3D reconstruction for crop mapping and yield estimation", 2010 11th International Conference on Control Automation Robotics & Vision, 2010. Disponible: 10.1109/icarcv.2010.5707436 [Accedido 20 agosto 2019].

[24] D. Andritoiu, L. Bazavan, F. Besnea, H. Roibu and N. Bizdoaca, "Agriculture autonomous monitoring and decisional mechatronic system", 2018 19th International Carpathian Control Conference (ICCC), 2018. Disponible: 10.1109/carpathiancc.2018.8399635 [Accedido 04 septiembre 2019].

[25] H. Cho, E. Kim and S. Kim, "Indoor SLAM application using geometric and ICP matching methods based online features", Robotics and Autonomous Systems, vol. 100, pp. 206-224, 2018. Disponible: 10.1016/j.robot.2017.11.011 [Accedido 14 febrero 2020].

[26] J. Zhang, S. Liu, B. Gao and C. Zhong, "An Improvement Algorithm for OctoMap Based on RGB-D SLAM", 2018 Chinese Control And Decision Conference (CCDC), 2018. Disponible: 10.1109/ccdc.2018.8407999 [Accedido 18 febrero 2021].

[27] S. Kajal. (2017). Improved visual SLAM: a novel approach to mapping and localization using visual landmarks in consecutive fotogramas, In Multimed Tools Appl, 2018. pp.7955–7958

[28] H. He, Y. Jia and L. Sun, "Simultaneous Location and Map Construction Based on RBPF-SLAM Algorithm", 2018 Chinese Control And Decision Conference (CCDC), 2018. Disponible: 10.1109/ccdc.2018.8407981 [Accedido 10 abril 2020].

[29] E. Hamuda, B. Mc Ginley, M. Glavin and E. Jones, "Automatic crop detection under field conditions using the HSV colour space and morphological operations", Computers and Electronics in Agriculture, vol. 133, pp. 97-107, 2017. Disponible: 10.1016/j.compag.2016.11.021 [Accedido 12 abril 2020].

[30] D. M. Woebbecke, G. E. Meyer, K. Von Bargen and D. A. Mortensen, "Shape Features for Identifying Young Weeds Using Image Analysis", Transactions of the ASAE, vol. 38, no. 1, pp. 271-281, 1995. Disponible: 10.13031/2013.27839 [Accedido 22 julio 2020].

[31] G. Meyer and J. Neto, "Verification of color vegetation indices for automated crop imaging applications", Computers and Electronics in Agriculture, vol. 63, no. 2, pp. 282-293, 2008. Disponible: 10.1016/j.compag.2008.03.009 [Accedido 22 julio 2020].

[32] J. Guerrero, G. Pajares, M. Montalvo, J. Romeo and M. Guijarro, "Support Vector Machines for crop/weeds identification in maize fields", Expert Systems with Applications, vol. 39, no. 12, pp. 11149-11155, 2012. Disponible: 10.1016/j.eswa.2012.03.040 [Accedido 25 julio 2020].

[33] M. Guijarro, G. Pajares, I. Riomoros, P. Herrera, X. Burgos-Artizzu and A. Ribeiro, "Automatic segmentation of relevant textures in agricultural images", Computers and Electronics in Agriculture, vol. 75, no. 1, pp. 75-83, 2011. Disponible: 10.1016/j.compag.2010.09.013 [Accedido 31 julio 2020].

[34] X. Burgos-Artizzu, A. Ribeiro, M. Guijarro and G. Pajares, "Real-time image processing for crop/weed discrimination in maize fields", Computers and Electronics in Agriculture, vol. 75, no. 2, pp. 337-346, 2011. Disponible: 10.1016/j.compag.2010.12.011 [Accedido 02 septiembre 2020].

[35] L. Tian and D. Slaughter, "Environmentally adaptive segmentation algorithm for outdoor image segmentation", Computers and Electronics in Agriculture, vol. 21, no. 3, pp. 153-168, 1998. Disponible: 10.1016/s0168-1699(98)00037-4 [Accedido 10 octubre 2020].

[36] L. Zheng, J. Zhang and Q. Wang, "Mean-shift-based color segmentation of images containing green vegetation", Computers and Electronics in Agriculture, vol. 65, no. 1, pp. 93-98, 2009. Disponible: 10.1016/j.compag.2008.08.002 [Accedido 10 octubre 2020].

[37] L. Zheng, D. Shi and J. Zhang, "Segmentation of green vegetation of crop canopy images based on mean shift and Fisher linear discriminant", Pattern Recognition Letters, vol. 31, no. 9, pp. 920-925, 2010. Disponible: 10.1016/j.patrec.2010.01.016 [Accedido 10 octubre 2020].

[38] W. Guo, U. Rage and S. Ninomiya, "Illumination invariant segmentation of vegetation for time series wheat images based on decision tree model", Computers and Electronics in Agriculture, vol. 96, pp. 58-66, 2013. Disponible: 10.1016/j.compag.2013.04.010 [Accedido 10 octubre 2020].

[39] X. Bai et al., "Vegetation segmentation robust to illumination variations based on clustering and morphology modelling", Biosystems Engineering, vol. 125, pp. 80-97, 2014. Disponible: 10.1016/j.biosystemseng.2014.06.015 [Accedido 21 octubre 2020].

[40] L. Xu, Y. Song, W. Zhang, Y. An, Y. Wang and H. Ning, "An efficient foreign objects detection network for power substation", Image and Vision Computing, vol. 109, p. 104159, 2021. Disponible: 10.1016/j.imavis.2021.104159 [Accedido 21 octubre 2020].

[41] C. Li, Z. Qu, S. Wang and L. Liu, "A method of cross-layer fusion multi-object detection and recognition based on improved faster R-CNN model in complex traffic environment", Pattern Recognition Letters, vol. 145, pp. 127-134, 2021. Disponible: 10.1016/j.patrec.2021.02.003 [Accedido 21 octubre 2020].

[42] M. García-Domínguez, C. Domínguez, J. Heras, E. Mata and V. Pascual, "UFOD: An AutoML framework for the construction, comparison, and combination of object detection models", Pattern Recognition Letters, vol. 145, pp. 135-140, 2021. Disponible: 10.1016/j.patrec.2021.01.022 [Accedido 21 octubre 2020].

[43] Techzizou. (2021, febrero, 23). YOLOv4 VS YOLOv4-tiny [Online]. Disponible: from https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny-97932b6ec8ec

[44] A. Bochkovskiy, C. Wang and H. Liao, (2020, Abril, 23). YOLOv4: Optimal Speed and Accuracy of Object Detection [Online]. Disponible: https://arxiv.org/pdf/2004.10934v1.pdf

[45] A. Morbekar, A. Parihar and R. Jadhav, "Crop Disease Detection Using YOLO", 2020 International Conference for Emerging Technology (INCET), 2020. Disponible: 10.1109/incet49848.2020.9153986 [Accedido 10 marzo 2021].

[46] C. Fairchild and T. Harman, ROS robotics by example. Birmingham: Packt Publishing Ltd, 2016, pp. 13-14.

[47] "msg - ROS Wiki", Wiki.ros.org, 2021. [Online]. Disponible: http://wiki.ros.org/msg. [Accedido 10 septiembre 2021].

[48] "Topics - ROS Wiki", Wiki.ros.org, 2021. [Online]. Disponible: http://wiki.ros.org/msg. [Accedido: 10 septiembre 2021].

[49] A. Koubaa, Robot operating systems (ros) - the complete reference. Springer, 2016, pp. 1-9.

[50] "Grayscale to RGB Conversion", Tutorialspoint.com, 2021. [Online]. Disponible: https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm. [Accedido 09 Noviembre 2021].

[51] "rgb2gray", https://www.mathworks.com, 2021. [Online]. Disponible: https://www.mathworks.com/help/matlab/ref/rgb2gray.html. [Accedido 09 Noviembre 2021].

[52] "Types of Morphological Operations", mathWorks, 2021. [Online]. Disponible: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html. [Accedido 23 Octubre 2021].

[53] R. Gonzales, R. Woods and S. Eddins, Digital image processing using MATLAB, 2nd ed. Berkshire: McGraw Hill, 2009, pp. 491,492,497,499.

[54] M. Sekhon, "Image Filters in Python", towards data science, 2021. [Online]. Disponible: https://towardsdatascience.com/image-filters-in-python-26ee938e57d2. [Accedido 12 Noviembre 2021].

[55] A. Shah. (2018, Junio 17). "Through The Eyes of Gabor Filter". [Online]. Disponible: https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97

[56] L. Moraru, C. Obreja, N. Dey and A. Ashour, "Dempster-Shafer Fusion for Effective Retinal Vessels' Diameter Measurement", Soft Computing Based Medical Image Analysis, pp. 149-160, 2018. Disponible: 10.1016/b978-0-12-813087-2.00008-7 [Accedido 12 Noviembre 2021].

[57] "imhist", mathworks.com, 2021. [Online]. Disponible: https://www.mathworks.com/help/images/ref/imhist.html. [Accedido 09 Noviembre 2021].

[58] "entropyfilt", mathworks, 2021. [Online]. Disponible: https://www.mathworks.com/help/images/ref/entropyfilt.html#mw_2347bab0-aec8-47a0-9664-d045c09822b1. [Accedido 09 Noviembre 2021].

[59] "Texture Analysis Using the Gray-Level Co-Occurrence Matrix (GLCM)- MATLAB & Simulink- MathWorks United Kingdom", Uk.mathworks.com, 2021. [Online]. Disponible: https://uk.mathworks.com/help/images/texture-analysis-using-the-gray-level-co-occurrence-matrix-glcm.html. [Accedido: 12 marzo 2021].

[60] M. Hall-Beyer, "GLCM Texture: A Tutorial v. 3.0 March 2017", Dx.doi.org, 2021. [Online]. Disponible: http://dx.doi.org/10.11575/PRISM/33280. [Accedido 12 marzo 2021].

[61] Universidad de Sevilla. Métodos estadísticos basados en la transformada Wavelet [Online]. Disponible: http://bibing.us.es/proyectos/abreproy/11494/fichero/PROYECTO%252FCapitulo+5.pdf

[62] M. Oliveira, V. Santos and A. Sappa, "Multimodal inverse perspective mapping", Information Fusion, vol. 24, pp. 108-121, 2015. Disponible: 10.1016/j.inffus.2014.09.003 [Accedido 15 marzo 2021].

[63] CV-Tricks.com. Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN, YOLO, SSD [Online]. Disponible: https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/.

[64] "¿Qué es una API de REST?", 2020. [Online]. Disponible: https://www.redhat.com/es/topics/api/what-is-a-rest-api. [Accedido 11 septiembre 2021].