Control de un brazo manipulador mediante visión por computadora para el monitoreo de enfermedades en las hojas de plantas de tomate

Andres Felipe Martinez Ordoñez

Trabajo de Grado para optar al título de Ingeniero Mecánico

Director

Carlos Alberto Flórez Arias

Candidato a Doctor en Ingeniería Mecánica

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería Mecánica

Bucaramanga

2026

Dedicatoria

A la memoria de mi padre, Raúl, quien me inspiró a superarme; su legado y enseñanzas me acompañarán siempre. A mi madre, Odilia, por su amor infinito y su entrega total; verla feliz es lo que me impulsa a seguir mejorando. A mis hermanos, Juan, Yessenia y Davinson, por su apoyo constante y su paciencia; este triunfo es tan mío como de ustedes.

Agradecimientos

Deseo expresar mi gratitud a Dios por permitirme culminar este proceso y obtener el título de ingeniero mecánico. A mi familia, gracias por su amor y apoyo incondicional en cada etapa de este largo camino. A mis amigos, por las risas y momentos que me permitieron despejar la mente cuando lo necesite. A Carlos Flórez, mi director de tesis, por sus valiosas enseñanzas y la paciencia demostrada durante el desarrollo de este proyecto de grado. De igual manera, agradezco a las Residencias Universitarias RESIUIS y a las personas que allí conocí, por convertirse en mi segundo hogar y mi familia. Finalmente, mi reconocimiento a la Universidad Industrial de Santander y a sus profesores por la excelente formación que me brindaron en el ámbito personal y profesional.

Tabla de Contenido

pág.

Introducción 13

1. Objetivos 16

2. Marco Referencial 17

2.1 Antecedentes 17

2.2 Marco Teórico 19

2.2.1 Robótica de manipuladores 19

2.2.2 Robot Operating System 19

2.2.3 El Modelo de Cámara Estenopeica 20

2.2.4 Procesamiento de nubes de puntos 22

2.2.5 Aprendizaje profundo para el análisis de imágenes 23

2.2.6 Planificación de Movimiento 26

3. Metodología 30

3.1 Entorno de Simulación y Herramientas 30

3.1.1 Plataforma de software 30

3.1.2 Diseño del mundo virtual 30

3.1.3 Modelo del Robot 31

3.2 Segmentación de la Hoja de Interés: Desarrollo y Entrenamiento del Modelo 32

3.2.1 Creación y Anotación del Conjunto de Datos (Dataset) 33

3.2.2 Entrenamiento del Modelo Mask R-CNN 35

3.3 Arquitectura del Sistema de Percepción 3D 36

3.3.1 Captura y Sincronización de Datos (capture_node) 38

3.3.2 Segmentación de la Hoja de Interés (segmentation_node) 38

3.3.3 Generación de la Nube de Puntos (cloud_processing_node) 39

3.3.4 Extracción de la Pose y el Vector Normal (normal_vector_extraction_node) 39

3.4 Sistema de Control y Planificación de Movimiento 41

3.4.1 Interfaz de Control del Robot (ur5e_controller_node) 41

3.4.2 Orquestación de la Tarea de Inspección (task_orchestrator_node) 42

3.5 Clasificación de Enfermedades 43

4. Resultados 44

4.1 Simulación del entorno en Gazebo 44

4.2 Sistema de Percepción 44

4.3 Validación del Sistema Integrado 50

5. Recomendaciones y Trabajos Futuros 54

6. Conclusiones 55

Referencias Bibliográficas 56

Apéndices 64

Lista de Figuras

pág.

Figura 1.	Esquema comunicación en ROS	20
Figura 2.	Modelo de la cámara estenopeica	21
Figura 3.	UR5e cobot	32
Figura 4.	Arquitectura del pipeline de percepción 3D	37
Figura 5.	Entorno de Simulación Integrado en Gazebo	45
Figura 6.	Curvas de entrenamiento y validación	46
Figura 7.	Hojas segmentadas con el modelo entrenado	48
Figura 8.	Secuencia completa del pipeline de percepción 3D	49
Figura 9.	Error de orientación por zona de inspección	52
Figura 10.	Tasa de éxito del pipeline por zona de inspección	53
Figura 11.	Distribución temporal de las fases del pipeline	54

Lista de Tablas

pág.

Tabla 1. Resultados de la evaluación del modelo Mask-RCNN 46

Tabla 2. Resultados de prueba del sistema integrado 51

Lista de Apéndices

pág.

Apéndice A.	Repositorio de GitHub	64
Apéndice B.	Dataset y script de entrenamiento de Mask R-CNN	65

Glosario

Backbone: es el componente de la red neuronal encargado de extraer características importantes como formas, bordes y texturas de las imágenes.

Batch size: cantidad de muestras de entrenamiento que se pasan a un modelo en una iteración completa.

FEM: o máquina de estados finitos es un sistema con un número establecido de estados, que solo puede estar en un estado a la vez y debe cumplir una condición para pasar a otro estado.

Framework: es una estructura que proporciona herramientas para crear proyectos de forma organizada, fácil y rápida.

GPU: o unidad de procesamiento gráfico, es un circuito electrónico que permite realizar gran cantidad de cálculos de forma simultánea gracias a sus múltiples núcleos de procesamiento.

IoU: es la métrica que mide la superposición entre dos regiones al dividir el área de intersección sobre el área de unión.

Learning rate: es el parámetro que determina que tan rápido un modelo aprende durante su entrenamiento.

mAP: es la métrica utilizada para medir el desempeño de una red neuronal en detección y segmentación. Compara las predicciones correctas, con la cantidad de objetos que detecta.

NLP: o procesamiento de lenguaje natural es la rama de la IA que enseña a las computadoras a entender y generar el lenguaje humano.

Octomap: es una representación tridimensional del espacio utilizando cubos, dice en que partes está libre u ocupado y con qué probabilidad.

Pipeline: es una secuencia de pasos organizados e interconectados para realizar una tarea específica.

Plugin: es un complemento que agrega una funcionalidad extra a un programa sin alterar el código fuente.

Pose: es una descripción de la posición (x, y, z) y orientación (cuaterniones) de un objeto en el espacio.

RGB-D: es una imagen que proporciona información de color rojo, verde y azul e información de profundidad de cada pixel que la conforma.

Resumen

Título: Control de un brazo manipulador mediante visión por computadora para el monitoreo de enfermedades en las hojas de plantas de tomate. [*]

Autor: Andres Felipe Martinez Ordoñez [**]

Palabras Clave: ROS2, Red neuronal, Visión de computadora, Brazo manipulador, Tomate.

Descripción: La agricultura moderna demanda métodos eficientes para la detección temprana de enfermedades, un factor crítico para la seguridad alimentaria. Esta tesis presenta el diseño y simulación de un sistema robótico autónomo para el monitoreo de enfermedades en hojas de plantas de tomate, abordando las limitaciones de la inspección manual. Desarrollado en el entorno ROS 2 Humble, el sistema integra un brazo manipulador Universal Robots UR5e de 6 DOF, equipado con una cámara RGB-D. El flujo de trabajo comienza con la segmentación de la hoja de interés mediante una red neuronal Mask R-CNN. A partir de los datos de profundidad y la máscara resultante, se genera una nube de puntos 3D de la hoja. Sobre esta nube se calcula su centroide y vector normal, definiendo una pose de inspección. Esta pose objetiva es enviada al framework de planificación MoveIt 2, que calcula y ejecuta una trayectoria libre de colisiones para posicionar el efector final del robot. Finalmente, un modelo de aprendizaje profundo (TOLD2) analiza la imagen capturada para diagnosticar enfermedades como el tizón temprano y tardío. El proyecto validó en simulación un flujo completamente automatizado, sentando una base sólida para futuras implementaciones.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Mecánicas. Director: Carlos Alberto Flólez Arias.

Abstract

Title: Control of a manipulator arm using computer vision for monitoring diseases in tomato plant leaves. [*]

Author: Andres Felipe Martinez Ordoñez [**]

Keywords: ROS2, Neural network, Computer vision, Manipulator arm, Tomato.

Description: Modern agriculture demands efficient methods for the early detection of diseases, a critical factor for food security. This thesis presents the design and simulation of an autonomous robotic system for monitoring diseases on tomato plant leaves, addressing the limitations of manual inspection. Developed in the ROS 2 Humble environment, the system integrates a 6-DOF Universal Robots UR5e manipulator arm equipped with an RGB-D camera. The workflow begins with the segmentation of the leaf of interest using a Mask R-CNN neural network. From the depth data and the resulting mask, a 3D point cloud of the leaf is generated. Its centroid and normal vector are calculated from this cloud, defining an inspection pose. This target pose is sent to the MoveIt 2 planning framework, which calculates and executes a collision-free trajectory to position the robot's end-effector. Finally, a deep learning model (TOLD2) analyzes the captured image to diagnose diseases such as early and late blight. The project validated a fully automated workflow in simulation, laying a solid foundation for future implementations.

---

[*] Bachelor Thesis

[**] Faculty of Physical-Mechanical Engineering. School of Mechanical Engineering. Director: Carlos Alberto Flólez Arias.

Introducción

Garantizar la seguridad alimentaria y alcanzar el objetivo de "hambre cero"para una población mundial proyectada en 9.8 mil millones de personas para 2050 constituye uno de los retos prioritarios de nuestro tiempo. Este desafío implica la necesidad de duplicar la producción de alimentos en una superficie agrícola limitada (Oliveira et al., 2021). Como respuesta, la agricultura intensiva en invernaderos se ha consolidado como una estrategia clave, pues permite desacoplar la producción de las estaciones y proteger los cultivos de eventos climáticos adversos. No obstante, esta ventaja viene acompañada de un desafío inherente: el ambiente controlado y estable de los invernaderos, si bien es óptimo para el crecimiento de las plantas, también resulta ser un caldo de cultivo idóneo para la rápida proliferación de patógenos fúngicos, bacterianos y virales (Schor et al., 2016).

La gravedad de esta amenaza se evidencia en las cifras de la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), que estima pérdidas anuales de entre el 20 % y el 40 % del rendimiento global de los cultivos a causa de plagas y enfermedades, incluso con la aplicación de millones de toneladas de pesticidas (King, 2017). En consecuencia, la capacidad de detectar estas enfermedades en sus etapas iniciales es un factor crítico para la protección de la cosecha y para la sostenibilidad del sistema agrícola. Un diagnóstico temprano y preciso permite acciones de control localizadas, minimizando el uso de agroquímicos y sus secuelas ambientales.

Frente a este panorama, los métodos tradicionales de monitoreo, basados en la inspección visual por parte de agricultores y agrónomos, resultan insuficientes. Esta metodología es esen-

cial, pero es lenta, subjetiva, intensiva en mano de obra y de alcance limitado en operaciones a gran escala. La consecuencia más directa es un diagnóstico tardío que, a menudo, obliga al uso indiscriminado y reactivo de fungicidas y pesticidas, con el consiguiente impacto negativo en la rentabilidad, el medio ambiente y la salud del consumidor (Khan et al., 2025).

En este contexto, la Agricultura de Precisión emerge como un paradigma transformador que, impulsado por los avances en robótica, visión por computadora, Internet de las Cosas (IoT) e inteligencia artificial, busca gestionar los cultivos planta por planta. Esta aproximación tecnológica ha dado lugar al desarrollo de robots agrícolas capaces de realizar tareas como la siembra, el deshierbe selectivo y la cosecha de manera autónoma (Oliveira et al., 2021). Específicamente para el monitoreo sanitario, se han desarrollado plataformas móviles que pueden patrullar los surcos de un cultivo de forma sistemática. No obstante, la simple navegación y captura de imágenes desde una posición fija resulta insuficiente para un diagnóstico preciso. La clave para una detección fiable de enfermedades reside en la adquisición de imágenes detalladas y bien orientadas de las hojas individuales, dado que es precisamente en el follaje donde primero se manifiestan los síntomas visuales característicos de la mayoría de patologías vegetales.

Un invernadero de tomates es un entorno 3D complejo y no estructurado. El reducido espacio entre plantas, las variaciones de altura y tamaño, la iluminación cambiante y la constante oclusión entre hojas y tallos hacen que obtener una imagen clara y bien orientada sea una tarea difícil. Una cámara fija en un robot móvil no puede garantizar la calidad ni el ángulo de visión necesarios. Para superar este obstáculo, se requiere un sistema capaz no solo de identificar una hoja, sino también de interactuar físicamente con el entorno para posicionar un sensor de manera

óptima.

La contribución principal de esta tesis es, por tanto, el diseño y la simulación de un sistema robótico que aborda este desafío. Se propone el control de un brazo manipulador de 6 grados de libertad mediante visión por computadora para la inspección y el monitoreo de enfermedades en hojas de tomate. El sistema automatiza toda la cadena de procesamiento: desde la segmentación de la hoja y la estimación de su pose, hasta la planificación y ejecución de un movimiento libre de colisiones para un posicionamiento preciso del efector final, culminando con la clasificación de la enfermedad mediante una red neuronal convolucional. Este trabajo demuestra la viabilidad de una solución robótica para la adquisición autónoma de datos en entornos agrícolas.

1. Objetivos

Objetivo general

Controlar un brazo manipulador de 6 DOF mediante visión por computadora para el monitoreo de enfermedades en las hojas de plantas de tomate en un ambiente simulado en ROS.

Objetivos específicos

Seleccionar y modelar el brazo manipulador y la planta de tomate en el entorno virtual.

Desarrollar la estrategia para determinar la posición, orientación y vector normal de una hoja a partir de las imágenes de una cámara RGB-D.

Crear un algoritmo para el posicionamiento del efector final del robot utilizando la librería OMPL en MoveIt.

Construir un nodo en ROS para el tratamiento de imágenes que permita utilizar un modelo de aprendizaje profundo preentrenado en la base de datos PlantVillage para la clasificación de enfermedades en las hojas de tomate.

2. Marco Referencial

2.1 Antecedentes

La aplicación de la robótica en la agricultura de precisión para el monitoreo de cultivos es un campo de investigación en plena expansión (Jin y Han, 2024), impulsado por la necesidad de automatizar tareas laboriosas y mejorar la eficiencia productiva. Esta sección revisa los trabajos más relevantes que conforman el estado del arte en la detección de enfermedades mediante sistemas robóticos, centrándose en las estrategias de percepción y manipulación necesarias para la adquisición de datos en entornos complejos.

El enfoque más directo para la automatización del monitoreo ha sido el uso de plataformas móviles equipadas con sensores. Trabajos como el de (Ouyang et al., 2022) proponen un robot móvil modular que se desplaza por los pasillos de un invernadero para la recolección masiva de imágenes. Si bien estos sistemas son eficaces para obtener una visión general del cultivo, presentan limitaciones significativas frente a la oclusión, donde unas hojas tapan a otras, y la incapacidad de posicionar la cámara en ángulos específicos para una inspección detallada.

Para superar estas barreras, la investigación ha evolucionado hacia el uso de brazos manipuladores que permiten una interacción activa con el entorno. (Schor et al., 2016) presenta uno de los trabajos pioneros en esta área, utilizando un brazo robótico industrial para la detección combinada de Oídio (Powdery Mildew) y el Virus del Moteado del Tomate (TSWV) en plantas de pimiento. Su investigación demuestra que diferentes enfermedades requieren distintas poses de inspección; y concluye que, para la detección temprana del Oídio, es imprescindible inspeccionar el envés de

la hoja, lo cual requiere que esta sea expuesta activamente, ratificando así una brecha clave en las capacidades de los sistemas de percepción pasiva.

Siguiendo esta línea de investigación, el trabajo de (Martin et al., 2021) presenta una arquitectura de control completa y genérica basada en ROS para un manipulador móvil destinado a la inspección y tratamiento de plagas. Su sistema, validado en campo, integra de forma desacoplada la navegación de la base móvil y la manipulación del brazo, utilizando MoveIt para el control de este último. Los autores reportan una tasa de fallo de casi el 50 % en la adquisición de imágenes válidas durante las pruebas de campo, atribuyendo estos fallos a la iluminación variable y, sobre todo, a las dificultades para posicionar el sensor de forma óptima con respecto a la hoja. Este resultado evidencia que el control inteligente de la pose del sensor sigue siendo un desafío abierto y una fuente de errores.

La solución a este desafío de posicionamiento reside en una percepción 3D que vaya más allá de la simple detección de la hoja. El trabajo de (Bao et al., 2018) aborda directamente el problema del sondeo de hojas. Proponen un pipeline donde, tras obtener una nube de puntos de alta precisión de la planta, segmentan las hojas y las dividen en parches planos (supervoxels). Posteriormente, utilizando el Análisis de Componentes Principales (PCA), calculan el vector normal a la superficie de estos parches para definir una pose de sondeo perpendicular. Este enfoque metodológico valida la estrategia de utilizar la geometría 3D de la hoja para guiar al manipulador.

2.2 Marco Teórico

2.2.1 Robótica de manipuladores

Un brazo manipulador se define como una cadena cinemática abierta, compuesta por una serie de eslabones (links) rígidos conectados mediante articulaciones (joints) móviles, culminando en un efector final que interactúa con el entorno (Craig, 2006). Para que un manipulador pueda posicionar un objeto de forma completa en el espacio tridimensional, es necesario definir tanto su posición (x,y,z) como su orientación (roll, pitch, yaw). Esto implica que el sistema debe contar con seis grados de libertad (DOF). Los grados de libertad representan las variables independientes de movimiento que posee un sistema mecánico; en un manipulador robótico, cada articulación contribuye con uno o más de estos grados, ya sea mediante movimientos rotacionales o lineales (Craig, 2006).

El análisis cinemático de un manipulador requiere la resolución de dos problemas fundamentales: la cinemática directa y la cinemática inversa. La cinemática directa consiste en determinar la posición y orientación del efector final a partir de los ángulos conocidos de las articulaciones. Por su parte, la cinemática inversa calcula los ángulos de las articulaciones necesarios para que el efector final llegue a una posición y orientación deseada. Resolver estos problemas es esencial para el control preciso del robot en tareas de manipulación (Craig, 2006).

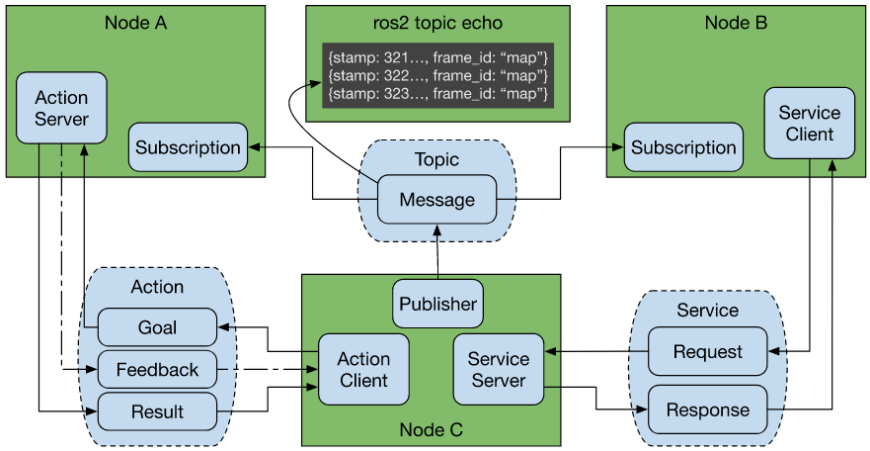
2.2.2 Robot Operating System

Es un "framework"de código abierto que proporciona herramientas, bibliotecas y convenciones que simplifican la creación de aplicaciones robóticas (Macenski et al., 2022). Dentro de

este entorno, los nodos constituyen los bloques fundamentales de ejecución, encargados de realizar tareas específicas dentro del sistema robótico; estos se comunican mediante tres mecanismos principales, como se muestra en la figura 1.

Figura 1

Esquema comunicación en ROS



Nota. Tomado de Macenski et al. (2022).

Los tópicos se usan para comunicación asíncrona tipo publicador/suscriptor, ideales en el envío continuo de datos como sensores o imágenes; servicios que permiten una comunicación síncrona de solicitud/respuesta para tareas puntuales que requieren confirmación; y acciones, utilizadas en tareas de larga duración, ya que permiten retroalimentación periódica y la opción de cancelación (Macenski et al., 2022).

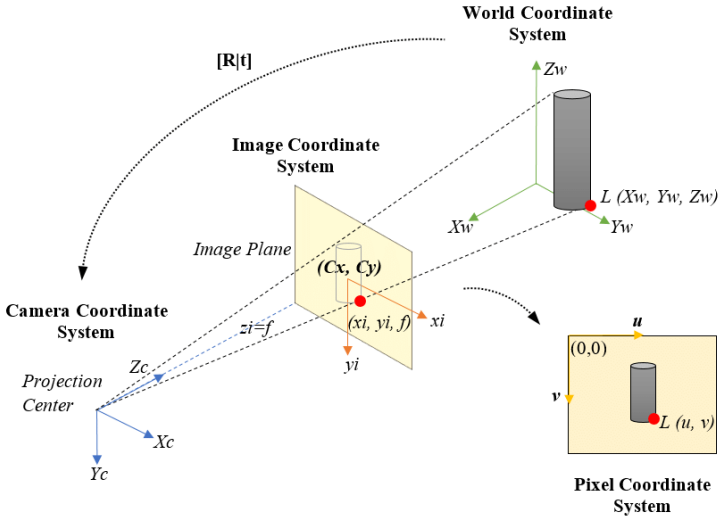
2.2.3 El Modelo de Cámara Estenopeica

La capacidad de transformar las mediciones de un sensor 2D, como una cámara, en una representación tridimensional del mundo es un principio fundamental de la percepción robótica.

Este proceso se fundamenta en el modelo de cámara estenopeica (o pinhole), una aproximación matemática que describe la geometría de la proyección de un punto 3D del mundo real sobre el plano 2D de la imagen, como se muestra en la figura 2.

Figura 2

Modelo de la cámara estenopeica



Nota. Tomado de Ortiz et al. (2017).

La relación matemática se expresa mediante la ecuación de proyección:

$$[u, v, 1]^T \approx K[R|t][X, Y, Z, 1]^T \tag{1}$$

Donde:

- (u, v) son las coordenadas en píxeles del punto en la imagen.
- (X, Y, Z) son las coordenadas del punto en el sistema de referencia del mundo.

- $[R|t]$ es la matriz extrínseca, que define la transformación de la pose de la cámara (rotación R y traslación t) con respecto al sistema del mundo.

- K es la matriz de parámetros intrínsecos, la cual es exclusiva de cada cámara y describe sus propiedades ópticas internas. Esta matriz tiene la forma:

$$K = [[fx, 0, cx], [0, fy, cy], [0, 0, 1]] \tag{2}$$

  Aquí, (fx, fy) representan la distancia focal de la cámara en unidades de píxeles, y (cx, cy) es el punto principal, que corresponde al centro óptico de la imagen. Esta matriz es fundamental, ya que mapea los puntos 3D desde el sistema de coordenadas de la cámara hacia el plano 2D de la imagen.

Una cámara RGB-D proporciona el valor de profundidad Z para cada píxel (u, v). Conociendo Z y los parámetros de la matriz intrínseca K, es posible despejar las coordenadas 3D del punto en el sistema de referencia de la cámara (Hartley y Zisserman, 2004).

2.2.4 Procesamiento de nubes de puntos

El filtrado de nubes de puntos se realiza para mejorar la calidad de los datos destinados a posteriores análisis geométricos. El filtrado estadístico elimina valores atípicos mediante el análisis de la distribución de distancias entre cada punto y sus vecinos. Los puntos cuya distancia promedio excede un umbral estadístico, típicamente basado en la desviación estándar, se consideran ruido y son descartados del conjunto de datos.

Una vez obtenida una nube limpia, es posible calcular características geométricas fundamentales como el centroide y el vector normal. El centroide se determina como la media aritmética de las coordenadas de todos los puntos. Para el cálculo del vector normal se emplea el Análisis de Componentes Principales (PCA), técnica estadística que permite reducir la dimensionalidad identificando las direcciones de máxima varianza. Para un conjunto de puntos que representa una superficie tridimensional, PCA identifica los ejes de mayor varianza mediante vectores propios (eigenvectors). El vector propio asociado al valor propio (eigenvalue) más pequeño corresponde a la dirección de mínima varianza y, para una superficie cuasi-planar, esta dirección es ortogonal a la superficie. Por consiguiente, dicho vector propio constituye una estimación del vector normal a la superficie (Bao et al., 2018).

2.2.5 Aprendizaje profundo para el análisis de imágenes

La visión por computadora es una disciplina que busca dotar a las máquinas de la capacidad de interpretar y comprender información visual del mundo real. Su objetivo principal radica en extraer conocimiento significativo de imágenes y videos mediante algoritmos computacionales, permitiendo a los sistemas automatizados realizar tareas que tradicionalmente requieren percepción humana.

El desarrollo de esta disciplina ha experimentado un avance significativo con la incorporación del deep learning, un subcampo del aprendizaje automático que utiliza redes neuronales artificiales con múltiples capas ocultas. Las redes neuronales, inspiradas en el funcionamiento del cerebro humano, procesan información a través de nodos interconectados que aprenden patrones complejos mediante el ajuste iterativo de pesos y sesgos (LeCun et al., 2015).

Dentro de las aplicaciones de visión por computadora, la segmentación de imágenes constituye una tarea fundamental que consiste en dividir una imagen en regiones semánticamente coherentes. La segmentación de instancias representa una variante más específica que no solo identifica objetos de diferentes clases, sino que también distingue entre instancias individuales de la misma clase, asignando etiquetas únicas a cada objeto detectado (Wang, 2016).

Paralelamente, la clasificación de imágenes busca asignar etiquetas categóricas a imágenes completas, determinando qué objetos o conceptos están presentes. Para estas tareas, las redes neuronales convolucionales (CNN) han demostrado ser particularmente efectivas debido a su capacidad de capturar características espaciales locales mediante filtros convolucionales, preservando la estructura bidimensional de las imágenes (Wang, 2016).

Mask R-CNN es una arquitectura de red neuronal convolucional que extiende la detección de objetos a la segmentación de instancias mediante un proceso de dos etapas. En la primera etapa, una Red de Propuestas de Regiones (RPN, por sus siglas en inglés) genera áreas candidatas que podrían contener objetos. En la segunda etapa, el modelo refina estas propuestas, realizando una clasificación y generando una máscara de segmentación a nivel de píxel para cada instancia detectada (He et al., 2018). Entre sus fortalezas destacan su elevada precisión, que resulta en máscaras detalladas y bien delimitadas, y su robustez para identificar objetos de diferentes tamaños. No obstante, su arquitectura secuencial de dos fases implica una velocidad de inferencia comparativamente menor.

YOLO (You Only Look Once) es una arquitectura de red neuronal convolucional concebida originalmente para la detección de objetos en tiempo real, cuyas iteraciones más recientes han in-

corporado capacidades de segmentación de instancias. A diferencia de los modelos de dos etapas, YOLO procesa la imagen en un único paso (single-pass), dividiéndola en una rejilla y prediciendo de forma simultánea las cajas delimitadoras, las clases de los objetos y sus correspondientes máscaras para cada celda. La principal ventaja de este enfoque reside en su excepcional velocidad de inferencia, una característica que, sin embargo, suele obtenerse a costa de una menor precisión en la delimitación de las máscaras (Redmon et al., 2016).

Los Transformers son una arquitectura de red neuronal cuyo funcionamiento se fundamenta en el mecanismo de atención, el cual permite relacionar cada parte de la entrada con las demás para comprender el contexto global. A diferencia de modelos anteriores, los Transformers procesan secuencias de datos de forma simultánea (en lugar de secuencial), lo que los hace altamente eficientes y permite el procesamiento en paralelo (Vaswani et al., 2023). Esta innovación ha impulsado el desarrollo de grandes modelos en áreas como el procesamiento de lenguaje natural (NLP), la visión artificial y la inteligencia artificial generativa.

El Segment Anything Model (SAM) es un modelo fundacional basado en la arquitectura Transformer, diseñado para la segmentación de imágenes a partir de indicaciones o prompts (tales como puntos, cajas delimitadoras o descripciones textuales). Su principal innovación es la capacidad de generalización para segmentar cualquier objeto de manera precisa sin requerir un entrenamiento específico para nuevas clases (zero-shot). Si bien las máscaras que produce son de excelente calidad, el modelo no efectúa clasificación de objetos y su operación depende de una entrada externa que suministre el prompt. Adicionalmente, su compleja arquitectura demanda recursos computacionales mayores (Kirillov et al., 2023).

El entrenamiento de estos modelos complejos frecuentemente emplea transfer learning, técnica que aprovecha conocimientos previos de modelos preentrenados en grandes conjuntos de datos. Posteriormente, mediante fine-tuning se ajustan los parámetros del modelo preentrenado para adaptarlo a tareas específicas, optimizando el rendimiento, disminuyendo el tiempo y los recursos computacionales requeridos.

2.2.6 Planificación de Movimiento

MoveIt2 es el framework de manipulación robótica más avanzado para ROS2, proporciona un ecosistema integral para la planificación, control y ejecución de movimientos en robots manipuladores. Este sistema ofrece herramientas sofisticadas para resolver los complejos desafíos asociados con el movimiento autónomo de brazos robóticos en entornos tridimensionales (Coleman et al., 2014).

El proceso de llevar un brazo robótico desde una configuración inicial hasta una pose objetivo comienza con la definición de la escena de planificación (Planning scene). Esta es una representación completa del entorno de trabajo que incluye obstáculos estáticos y dinámicos, el modelo cinemático del robot y las restricciones de seguridad. La escena actúa como el contexto espacial donde se evaluarán todas las posibles trayectorias, proporcionando información crítica sobre colisiones potenciales y limitaciones del espacio de trabajo (PickNik Robotics, 2025).

Una vez establecido el entorno, el sistema debe resolver el problema de cinemática inversa, que consiste en determinar los valores angulares de las articulaciones necesarios para alcanzar una pose cartesiana específica del efector final. Este problema puede tener múltiples soluciones o ninguna, dependiendo de la configuración deseada y las limitaciones físicas del robot. Los al-

goritmos de cinemática inversa en MoveIt2 utilizan métodos numéricos iterativos y soluciones analíticas cuando están disponibles, seleccionando configuraciones que minimicen singularidades y maximicen la maniobrabilidad.

El núcleo del proceso de planificación opera en el espacio de configuración (C-space), un espacio multidimensional donde cada dimensión representa el valor de una articulación del robot. En este espacio, el robot se reduce a un punto y los obstáculos se transforman en regiones prohibidas. La planificación de trayectorias se convierte entonces en encontrar un camino libre de colisiones desde el punto inicial hasta el punto objetivo dentro de este espacio de configuración (Lozano-Perez, 1983).

MoveIt2 ofrece tres familias principales de algoritmos de planificación de movimiento: planificadores basados en muestreo (OMPL), planificadores de optimización de trayectorias basados en gradientes (CHOMP) y planificadores de optimización con enfoque probabilístico (STOMP).

Los planificadores basados en optimización (CHOMP y STOMP) encuentran trayectorias que minimizan funciones de costo, generando movimientos suaves y cortos. Estos planificadores requieren un mayor tiempo de ejecución y son susceptibles a quedar atrapados en mínimos locales, lo que puede resultar en fallos en la planificación.

Los algoritmos basados en muestreo exploran el espacio de configuración del robot mediante la construcción aleatoria de "árboles o mapas"de estados libres de colisión hasta conectar las configuraciones inicial y final. Esta aproximación los hace particularmente robustos para problemas de alta dimensionalidad en entornos no estructurados y dinámicos (Orthey et al., 2023). Algunos de los algoritmos que destacan por su rendimiento son:

RRTConnect (Kuffner y LaValle, 2000): Emplea búsqueda bidireccional, construyendo simultáneamente dos árboles desde el inicio y la meta, intentando conectarlos en cada iteración (Kuffner y LaValle, 2000).

LBKPIECE (Lazy Bi-directional KPIECE): Utiliza una estrategia de exploración guiada por una rejilla proyectada del espacio de configuración, siendo particularmente eficaz en entornos muy congestionados. Su naturaleza "perezosa"pospone la validación de colisiones, optimizando la velocidad (Sucan y Kavraki, 2012).

SBL (Single-Query Bi-directional Lazy Planner): Combina la búsqueda bidireccional con la validación perezosa para acelerar la planificación (Sanchez-Ante y Latombe, 2002).

EST (Expansive Space Trees): Favorece el crecimiento del árbol hacia regiones menos exploradas del espacio de configuración, ofreciendo una alternativa a la selección del vecino más cercano de RRT (Phillips et al., 2004).

Diversos estudios comparativos (Liu y Liu, 2022; Orthey et al., 2023; Yang et al., 2023; Zhang et al., 2025) han evaluado algoritmos de planificación en múltiples escenarios con diferentes niveles de complejidad, dimensiones del espacio de configuración (2-31 grados de libertad), tipos de tareas (manipulación, navegación por pasajes estrechos, extensión de alcance, entre otras) y diversos manipuladores robóticos (UR5, Franka Emika, Panda, KUKA, Baxter, entre otros). Los resultados demuestran que RRTConnect, aunque no genera las trayectorias más suaves ni más cortas en comparación con los métodos de optimización, presenta ventajas significativas en términos de robustez y eficiencia: exhibe una tasa de éxito superior, tiempos de cómputo promedio más bajos y no requiere ajustes de configuración específicos para cada escenario (Liu y Liu, 2022; Orthey

et al., 2023; Yang et al., 2023; Zhang et al., 2025).

Una vez que el algoritmo de planificación encuentra una trayectoria válida, el proceso continúa con la ejecución del movimiento. MoveIt2 coordina con los controladores de bajo nivel para enviar referencias de posición, velocidad y par motor a cada articulación, monitoreando continuamente el progreso y ajustando la ejecución según sea necesario. Este proceso incluye mecanismos de seguridad que pueden detener o modificar el movimiento si se detectan condiciones imprevistas, garantizando operaciones seguras y precisas en aplicaciones críticas como el posicionamiento de cámaras para sistemas de visión por computadora (Coleman et al., 2014).

3. Metodología

Este capítulo detalla la metodología empleada para el diseño y simulación del sistema robótico de monitoreo. Se describen las herramientas de software utilizadas, la configuración del entorno de simulación, la arquitectura del sistema de percepción 3D y el flujo de trabajo integrado para la ejecución de la tarea de inspección.

3.1 Entorno de Simulación y Herramientas

3.1.1 Plataforma de software

El sistema se desarrolló sobre una base de software de código abierto, seleccionada por su robustez, amplio soporte comunitario y su uso frecuente en la investigación robótica. Todas las implementaciones se realizaron sobre la distribución ROS 2 Humble Hawksbill, que cuenta con Soporte a Largo Plazo (LTS). Se utilizó Gazebo (Ignition Fortress) como el simulador 3D, fue seleccionado por su estrecha integración con ROS 2, su capacidad para modelar la física de cuerpos rígidos de manera realista, y su soporte nativo para la simulación de una amplia gama de sensores, incluyendo las cámaras RGB-D utilizadas en este proyecto.

3.1.2 Diseño del mundo virtual

Los modelos 3D de la planta de tomate se generaron mediante un script en Python diseñado para ejecutarse en Blender; este script se obtiene del repositorio de GitHub fields-ignition (Polgár, 2025), cuyo objetivo es generar entornos de prueba para robots agrícolas.

La planta se estructura en torno a un tallo principal, modelado como una serie de entre 35 y 40 nodos conectados en el espacio 3D. El tallo se eleva verticalmente hasta una altura de 1.25

metros. El grosor del tallo decrece desde la base (con un radio de 9 mm, sujeto a una pequeña variación aleatoria) hasta la punta, donde se reduce hasta alcanzar el 42 % de su grosor inicial.

La trayectoria en espiral del tallo se logra al aplicar una rotación de 120 grados entre cada nodo, simulando un patrón de crecimiento natural. En cada nodo del tallo principal, a excepción del primero, se añade una rama secundaria que puede contener hojas, flores y frutos. La configuración de esta rama se basa en la posición relativa del nodo a lo largo del tallo, permitiendo simular diferentes estados de desarrollo según la altura.

Finalmente, los frutos aparecen en algunas ramas secundarias; su presencia y ubicación se determinan de manera probabilística. De este modo, el número y la distribución de los frutos varían en cada ejecución.

3.1.3 Modelo del Robot

Para la simulación se usó un robot manipulador Universal Robot UR5e como el que se muestra en la figura 3, un robot colaborativo de 6 grados de libertad (DOF); el robot cuenta con un alcance de 850 mm y una carga útil de 5 kg. El robot se instala sobre una base fija de 0.65 m de altura, que modela un caso de uso en una plataforma móvil.

La representación del robot en el ecosistema ROS se gestionó a través del paquete `ur5e_description`. Este paquete se basa en el paquete Universal_Robots_ROS2_Description disponible en el repositorio oficial de GitHub de Universal Robots; este contiene los archivos de descripción del robot en formato URDF (Unified Robot Description Format) y XACRO (XML Macros), que definen la estructura cinemática del manipulador, así como sus propiedades inerciales y las mallas geométricas para la visualización y la detección de colisiones. Además, en este paquete se

Figura 3

UR5e cobot

Nota. Tomado de Universal Robots A/S (2025).

integró el modelo de una cámara RGB-D en el efector final del robot y se configuraron los plugins de ros2_control necesarios para establecer la interfaz de control entre los planificadores de ROS 2 y las articulaciones simuladas en Gazebo.

3.2 Segmentación de la Hoja de Interés: Desarrollo y Entrenamiento del Modelo

Una máscara de calidad es fundamental para la extracción del vector normal de la hoja, por lo tanto, la característica principal del modelo de segmentación seleccionado debe ser su precisión. Estudios comparativos realizados por (Bae et al., 2025) y (Choi et al., 2024) en la segmentación

de revestimiento de túneles y la detección de grietas en concreto, demuestran mayor fiabilidad y precisión de Mask R-CNN sobre YOLOv8. Mask R-CNN fue superior en ambas tareas, logrando un IoU de 0.973 vs. 0.894 en revestimiento y de 96.5 vs. 90.6 en segmentación de grietas.

En experimentos realizados por (Yung et al., 2025) y (Williams et al., 2023) en segmentación de hojas de Atriplex lentiformis y hojas de papa, Mask R-CNN obtiene mayor precisión frente a SAM, demostrando que un modelo al que se le realizó fine-tunning es más eficaz para tareas especializadas que un modelo fundacional. Mask R-CNN superó a SAM en ambos casos, alcanzando una precisión de 0.881 frente a 0.692 en hojas de Atriplex, y de 74.7 frente a 60.3 en hojas de papa.

Si bien YOLOv destaca por su rapidez en entrenamiento e inferencia (Bae et al., 2025; Choi et al., 2024) siendo hasta 1.65 veces más veloz que Mask R-CNN según (Sapkota et al., 2024) y requiriendo menos de la mitad del tiempo de entrenamiento (Bae et al., 2025), para este proyecto se seleccionó Mask R-CNN como el modelo de segmentación debido a que ofrece un desempeño superior en la precisión de los resultados.

El proceso de entrenamiento de Mask R-CNN se dividió en dos fases: la creación y anotación del conjunto de datos, y el entrenamiento y configuración del modelo.

3.2.1 Creación y Anotación del Conjunto de Datos (Dataset)

Se construyó un dataset siguiendo un enfoque de anotación semi-automatizado.

1. Recolección de Datos: El dataset se creó a partir de la combinación de cuatro colecciones de imágenes de libre uso, obtenidas de la plataforma Roboflow: leaves_2_dataset (farm, 2024), tomato_village_fixed_dataset (mastersthesis, 2024), 3class-iduv1_dataset (Workspa-

ce, 2022) y tomato-leaf-object-detection_dataset (mastersthesis, 2024). A este conjunto de datos se agregaron 200 fotografías capturadas en el entorno de simulación, para un total de 1788 imágenes. Se buscó una alta variabilidad en las imágenes, incluyendo diferentes condiciones de iluminación, ángulos de cámara y diferentes etapas de crecimiento de la planta.

2. Anotación Semi-Automatizada: Para acelerar el proceso de etiquetado de máscaras de segmentación, se usó GroundingDINO para detectar las hojas en las imágenes con un enfoque zero-shot, el cual permite identificar objetos a partir de descripciones en lenguaje natural sin necesidad de entrenamiento adicional. Las cajas de detección obtenidas se emplearon como entrada para el modelo Segment Anything Model (SAM), que generó máscaras de segmentación detalladas de las hojas identificadas. Los parámetros clave para la detección se establecieron con umbrales de caja y texto igual a 0.45 para la clase 'leaf'. Se filtraron polígonos muy pequeños o demasiado grandes y se aplicó una aproximación del 60 % para suavizar los contornos del polígono de anotación.

3. Refinamiento y Exportación: Se utilizó la plataforma Roboflow para mejorar la calidad del dataset final. El proceso de depuración incluyó:

    - La corrección de máscaras imprecisas o con errores geométricos.
    - La eliminación de detecciones incorrectas (objetos que no eran hojas).
    - La fusión o eliminación de instancias donde una misma hoja había sido segmentada múltiples veces.

El dataset se dividió en un conjunto de entrenamiento de 1245 imágenes (70 %), un conjunto de validación de 364 imágenes (20 %) y un conjunto de prueba de 179 imágenes (10 %). Finalmente, el dataset completo se exportó en el formato COCO JSON, que es el formato estándar de entrada para la implementación de Mask R-CNN utilizada.

3.2.2 Entrenamiento del Modelo Mask R-CNN

El entrenamiento del modelo se llevó a cabo en la plataforma Kaggle, aprovechando su entorno de cómputo equipado con recursos informáticos potentes como GPUs y TPUs. El entrenamiento del modelo de segmentación se llevó a cabo utilizando el framework PyTorch. Se adoptó una estrategia de aprendizaje por transferencia, se partió de un modelo pre-entrenado MaskR-CNN_ResNet50_FPN disponible en la librería torchvision, aprovechando los pesos entrenados previamente en el dataset COCO para inicializar el backbone de la red. El entrenamiento se estructuró en dos fases secuenciales:

- Fase 1: Ajuste de las Capas Superiores: En la primera fase, se realizó un ajuste fino exclusivamente de las capas superiores de la red (clasificación y segmentación), manteniendo congelados los pesos del backbone ResNet50. Esta etapa se extendió por 5 épocas, utilizando una tasa de aprendizaje (learning rate) fija de 0.005. El objetivo de esta fase inicial fue adaptar las capas finales del modelo al dominio específico de las hojas de tomate sin desestabilizar las características genéricas ya aprendidas.

- Fase 2: Entrenamiento Integral: En la segunda fase, se desbloquearon todas las capas del modelo para su entrenamiento. Esta etapa se prolongó por un máximo de 30 épocas, con una

tasa de aprendizaje inicial de 0.0005. Se implementó un planificador de tasa de aprendizaje de tipo StepLR, configurado para reducir el learning rate en un factor de 0.1 cada 10 épocas, permitiendo un ajuste más fino de los pesos a medida que el modelo se acercaba a la convergencia.

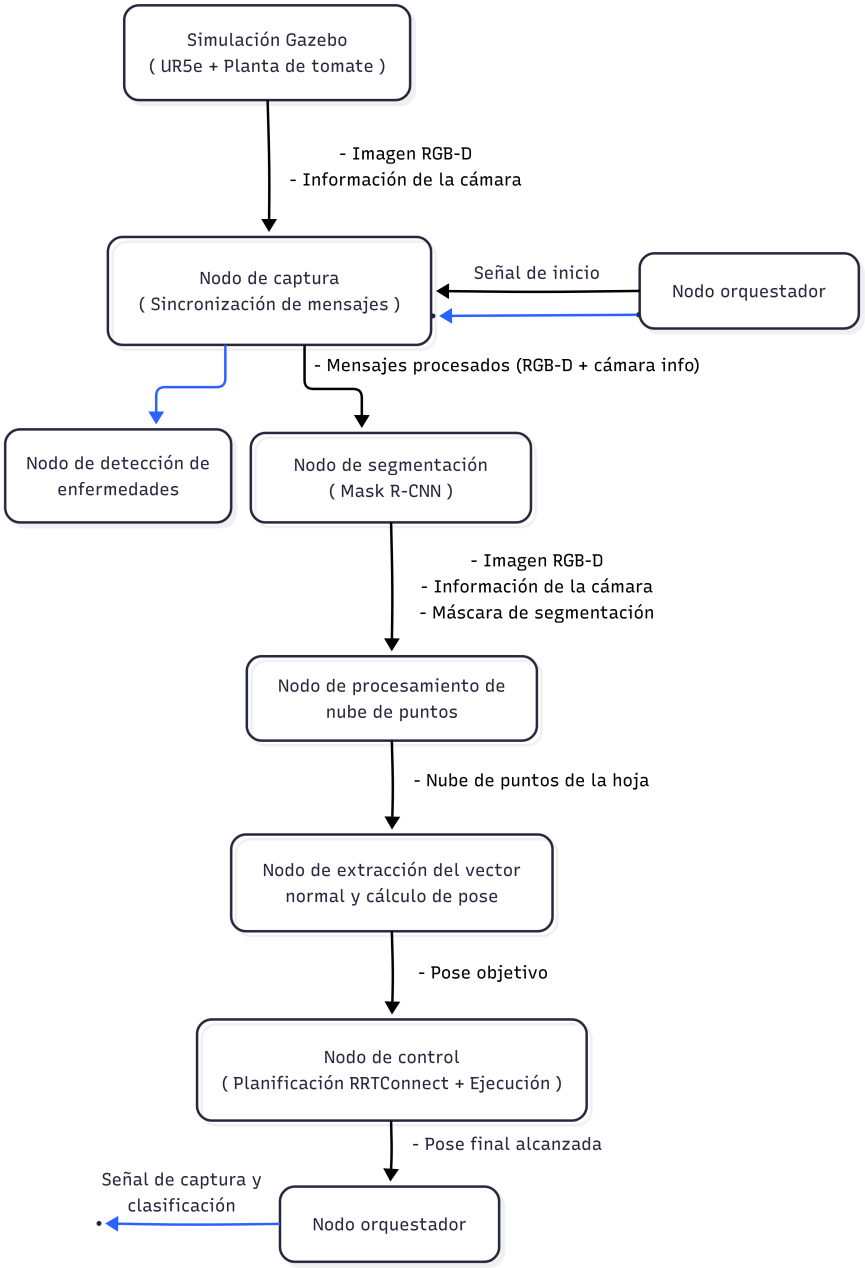
Para la alimentación de datos al modelo, se configuraron dos DataLoaders distintos. Para el conjunto de entrenamiento, se utilizó un tamaño de lote (batch_size) de 8, mientras que para el conjunto de validación se estableció un batch_size de 1. Finalmente, se implementó un mecanismo de parada temprana (early stopping). Este callback monitoreó la métrica de Precisión Media Promedio (mAP) sobre el conjunto de validación. El entrenamiento se detuvo automáticamente si esta métrica no mejoraba tras 5 épocas consecutivas; si esto sucedía, se restauraban los pesos del modelo correspondientes a la época con el mAP más alto.

3.3 Arquitectura del Sistema de Percepción 3D

Este subsistema es responsable de procesar los datos originales del sensor RGB-D para identificar, aislar y finalmente determinar la pose tridimensional de una hoja de interés. Toda esta funcionalidad se encapsuló en un paquete de ROS 2 modular y reutilizable denominado object_point_cloud_processor. El sistema de percepción fue diseñado como una cadena de procesamiento secuencial, donde la salida de un nodo sirve como entrada para el siguiente. En la figura 4 se observa la arquitectura completa del sistema de percepción 3D.

Figura 4

Arquitectura del pipeline de percepción 3D



A continuación, se detalla la implementación y función de cada uno de los nodos que componen esta arquitectura.

3.3.1 Captura y Sincronización de Datos (capture_node)

El nodo `capture_node` fue diseñado para suscribirse a los tópicos originales de la cámara RGB-D simulada (`/rgbd_camera/color/image_raw`, `/rgbd_camera/depth/image_raw` y `/rgbd_camera/color/camera_info`). Dada la naturaleza asíncrona de la publicación de datos en ROS, se implementó un sincronizador de mensajes para asegurar que el conjunto de la imagen RGB, la imagen de profundidad y los parámetros de la cámara correspondieran exactamente al mismo instante de tiempo.

La publicación de estos datos se hace cuando otro nodo los requiera, para esto el nodo dispone un servicio de tipo std_srvs/Trigger llamado `/trigger_capture_full`. Cuando un cliente llama al servicio, el capture_node publica el último conjunto de datos sincronizados; así, el sistema de percepción permanece inactivo mientras no se use.

3.3.2 Segmentación de la Hoja de Interés (segmentation_node)

Una vez que se publican los datos sincronizados, el segmentation_node aísla la hoja de interés usando una red neuronal Mask R-CNN entrenada para segmentar hojas de tomate. Al recibir la imagen RGB, el modelo la procesa y genera una máscara binaria para cada hoja detectada.

Para evitar la ambigüedad en escenas con múltiples hojas, se implementó una lógica para seleccionar la instancia con la mayor puntuación de confianza devuelta por el modelo. Una vez seleccionada la hoja principal, el nodo publica la imagen RGB original, la imagen de profundidad, la máscara binaria correspondiente a la hoja seleccionada y la información de la cámara en un nuevo conjunto de tópicos.

3.3.3 Generación de la Nube de Puntos (cloud_processing_node)

El nodo `cloud_processing_node` es el responsable de la transición del dominio 2D al 3D. El nodo se suscribe a las salidas del segmentation_node y realiza los siguientes pasos:

1. Enmascaramiento de la Profundidad: La máscara binaria se aplica a la imagen de profundidad. Los píxeles de la imagen de profundidad que no pertenecen a la hoja se descartan.

2. Retroproyección 3D: Utilizando la librería Open3D, se itera sobre cada píxel restante de la imagen de profundidad enmascarada. Cada píxel (u, v) con un valor de profundidad Z se proyecta a sus coordenadas 3D (X, Y, Z) en el sistema de referencia de la cámara, utilizando para ello la matriz de parámetros intrínsecos K obtenida del tópico camera_info.

3. Coloreado de Puntos: A cada punto 3D generado se le asigna el valor de color (R, G, B) del píxel correspondiente en la imagen RGB original.

El resultado es una nube de puntos 3D coloreada de la hoja de interés. Esta nube de puntos se empaqueta en un mensaje de ROS de tipo `sensor_msgs/PointCloud2` y se publica en el tópico /object_point_cloud.

3.3.4 Extracción de la Pose y el Vector Normal (normal_vector_extraction_node)

La etapa final del pipeline consiste en analizar la geometría de la nube de puntos para extraer una pose de inspección accionable por el robot. El nodo normal_vector_extraction_node realiza un análisis geométrico en varios pasos:

1. Pre-procesamiento: Se aplica un sub-muestreo por voxeles con un tamaño de voxel de 1 mm

para reducir la cantidad de puntos; posteriormente, se aplicó un filtro estadístico para eliminar el ruido y los puntos aislados. Finalmente, se utiliza el algoritmo de clustering DBSCAN (Density-Based Spatial Clustering of Applications with Noise) para agrupar los puntos en regiones densas y se selecciona el clúster con el mayor número de puntos, asumiendo que representa la superficie principal de la hoja.

2. Cálculo del centroide: Se calcula el centroide como el promedio de las coordenadas x, y, z del conjunto de puntos seleccionado.

3. Estimación de la normal: Se aplica el Análisis de Componentes Principales (PCA) sobre el clúster de puntos. El vector propio asociado al menor valor propio se toma como la estimación del vector normal a la superficie de la hoja.

4. La posición objetivo final se calculó desplazando el punto desde el centroide de la hoja 15 cm a lo largo de la dirección del vector normal. Simultáneamente, se calculó una orientación en formato de cuaternión que alinea el eje de visión de la cámara (eje Z) de forma antiparalela al vector normal, garantizando así una vista perpendicular a la superficie de la hoja. Esta pose completa, calculada inicialmente en el sistema de coordenadas de la cámara (rgbd_camera_link), fue transformada utilizando la librería TF2 de ROS al marco de referencia global (world).

La pose final transformada es publicada en el tópico /normal_extraction/target_pose, quedando disponible para que el sistema de control del robot la utilice como objetivo de su próximo movimiento.

3.4 Sistema de Control y Planificación de Movimiento

Basado en el análisis de la literatura realizado en el marco teórico, se concluyo que RRT-Connect es el algoritmo de planificación de movimiento más adecuado para el proyecto gracias a su alta tasa de éxito y su velocidad.

El control del brazo manipulador UR5e para alcanzar la pose de inspección calculada por el sistema de percepción se gestionó a través del framework MoveIt 2.

3.4.1 Interfaz de Control del Robot (ur5e_controller_node)

Para simplificar la interacción con MoveIt se desarrolló el nodo ur5e_controller_node. Este nodo envuelve la funcionalidad de la MoveGroupInterface de MoveIt y la expone a través de interfaces de acción de ROS 2. Se eligió el mecanismo de acción porque las tareas de movimiento son de larga duración y se benefician de la capacidad de proporcionar retroalimentación continua y de ser canceladas. El nodo implementó dos servidores de acción:

1. /ur5e/go_to_named_target: Este servidor de acción acepta una meta que contiene el nombre de una pose predefinida en el SRDF (ej., "home"). Al recibir la meta, el nodo utiliza MoveIt para planificar y ejecutar una trayectoria hacia dicha configuración articular.

2. /ur5e/move_to_pose: Este servidor de acción acepta una meta de tipo `geometry_msgs/PoseStamped`, que corresponde a una pose cartesiana específica para el efector final. Es la interfaz utilizada para mover el robot a la pose de inspección calculada por el sistema de percepción.

Para aumentar la posibilidad de encontrar una trayectoria válida hacia la pose de captura, se

agregó una función que primero planea una trayectoria con la orientación original, si falla, intenta planificar rotando la pose sobre el eje Z manteniendo así la perpendicularidad a la hoja, esto lo realiza cada 45° hasta alcanzar los 360°.

3.4.2 Orquestación de la Tarea de Inspección (task_orchestrator_node)

El nodo task_orchestrator_node coordina la interacción entre la percepción y el control para ejecutar la misión de inspección de forma autónoma. La lógica de control del orquestador se implementó como una Máquina de Estados Finitos (FSM). Los estados principales de la FSM se definieron de la siguiente manera:

- INITIALIZING: Estado inicial donde el nodo espera a que todos los servidores de acción (`/ur5e/go_to_named_target`, `/ur5e/move_to_pose`) y los servicios de captura (`/trigger_capture_full` y `/trigger_leaf_detection`) estén activos y disponibles.
- MOVING_TO_HOME: Una vez inicializado, el nodo pasa a este estado, donde envía una meta al servidor de acción `/ur5e/go_to_named_target` para mover el robot a su posición de inspección.
- TRIGGERING_FULL_CAPTURE: Al completarse con éxito el movimiento a pose de inspección, el nodo llama al servicio `/trigger_capture_full` para iniciar el pipeline de percepción 3D.
- WAITING_FOR_POSE: En este estado, el orquestador se suscribe al tópico `/normal_extraction/target_pose` y espera la publicación de la pose objetiva calculada por el sistema de percepción. Se implementó un temporizador de espera para manejar casos en los que la percepción

no logre encontrar un objetivo.

- MOVING_TO_TARGET: Al recibir una pose válida, el nodo envía esta pose como una meta al servidor de acción `/ur5e/move_to_pose`. El robot se mueve para alinear su efector final con la hoja de interés.

- TRIGGERING_LEAF_CAPTURE: Una vez que el robot ha alcanzado la pose objetivo, el nodo transita a este estado y llama al servicio `/trigger_leaf_detection`. Esta acción solicita la publicación de la imagen RGB para la clasificación de enfermedades.

- COMPLETED / FAILED: Estados finales que indican el resultado de la tarea. El estado FAILED se alcanza si alguna de las acciones de movimiento falla, si el servicio de percepción no responde, o si se excede el tiempo de espera para la pose objetivo.

3.5 Clasificación de Enfermedades

Se desarrolló el nodo disease_detector_node, este nodo realiza la inferencia sobre una imagen para determinar si la hoja está sana o presenta alguna enfermedad. El nodo carga una red neuronal convolucional ResNet50 entrenada en las clases Tomato_Early_blight, Tomato_Late_blight, Tomato_healthy y Tomato_mosaic_virus. Una vez que el robot alcanza la pose final, se captura la imagen y se publica en el topic /desease_detection_img, se clasifica por el modelo y se imprime

en la terminal la clase y su confianza.

4. Resultados

4.1 Simulación del entorno en Gazebo

El entorno de simulación se compuso de una planta de tomate modelada sobre un terreno agrícola, como se ilustra en la figura 5. La base del manipulador se situó a una distancia horizontal de 0.7 metros del tallo central de la planta. Esta decisión se fundamentó en las recomendaciones del fabricante del UR5e (Universal Robots A/S, 2025), que especifican un espacio de trabajo óptimo en un rango de 200 mm a 750 mm para evitar singularidades. A esta distancia, el 56 % de la planta, que tiene un radio de 0.425 m está contenido dentro de la zona de operación óptima del brazo robótico. El sensor RGB-D simulado fue configurado con una resolución de 640×480 píxeles, un campo de visión horizontal de 60° y una frecuencia de actualización de 20 Hz.

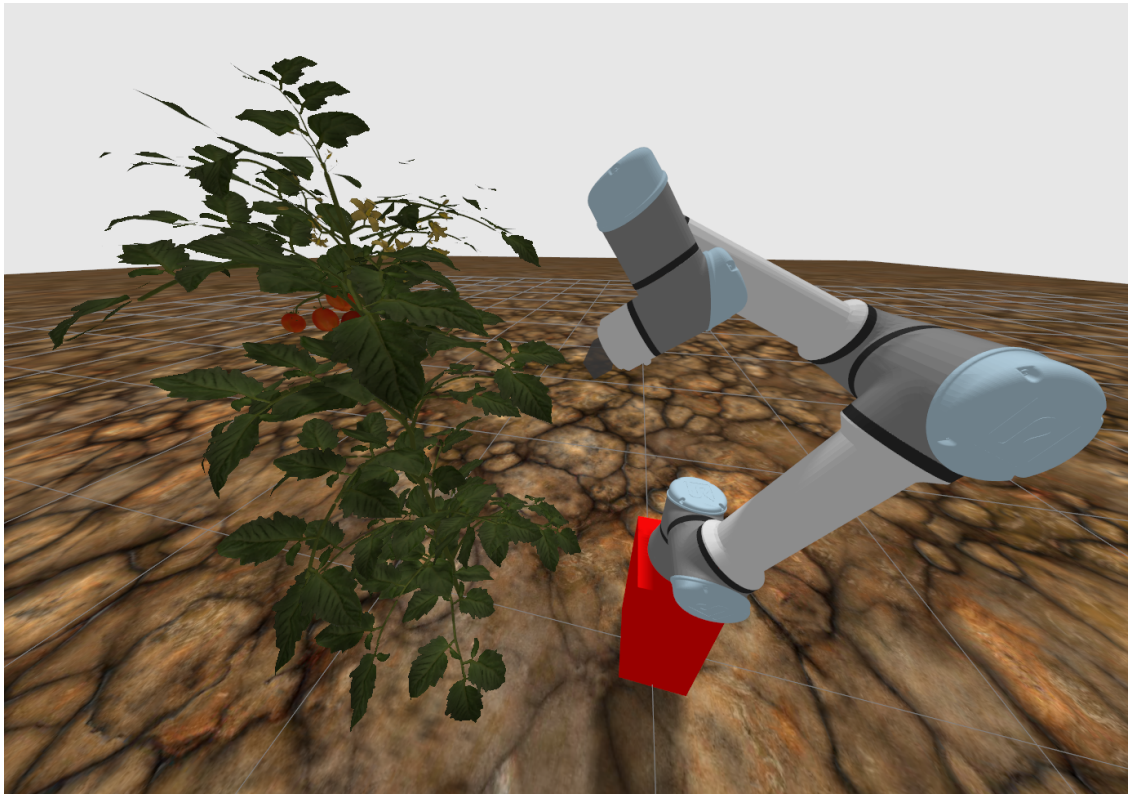
4.2 Sistema de Percepción

En la figura 6 se muestran las gráficas con los datos obtenidos durante el entrenamiento del modelo Mask-RCNN. Durante las 5 primeras etapas, la pérdida de entrenamiento comenzó alta (2.16) y descendió de manera constante hasta 1.6994, a medida que las capas superiores se ajustaban a la nueva distribución de datos. La pérdida de validación siguió una tendencia similar, disminuyendo de 1.72 a 1.64. Como fue esperado, la pérdida de validación fue menor que la de entrenamiento en estas primeras épocas. Por su parte, el mAP de Máscara (mAP@0.5:0.95) mostró un progreso modesto, pasando de 0.26 a 0.31, ya que la capacidad del modelo para extraer características estuvo limitada por el backbone congelado.

Al descongelar todas las capas y reducir la tasa de aprendizaje, se produjo una mejora: la

Figura 5

Entorno de Simulación Integrado en Gazebo

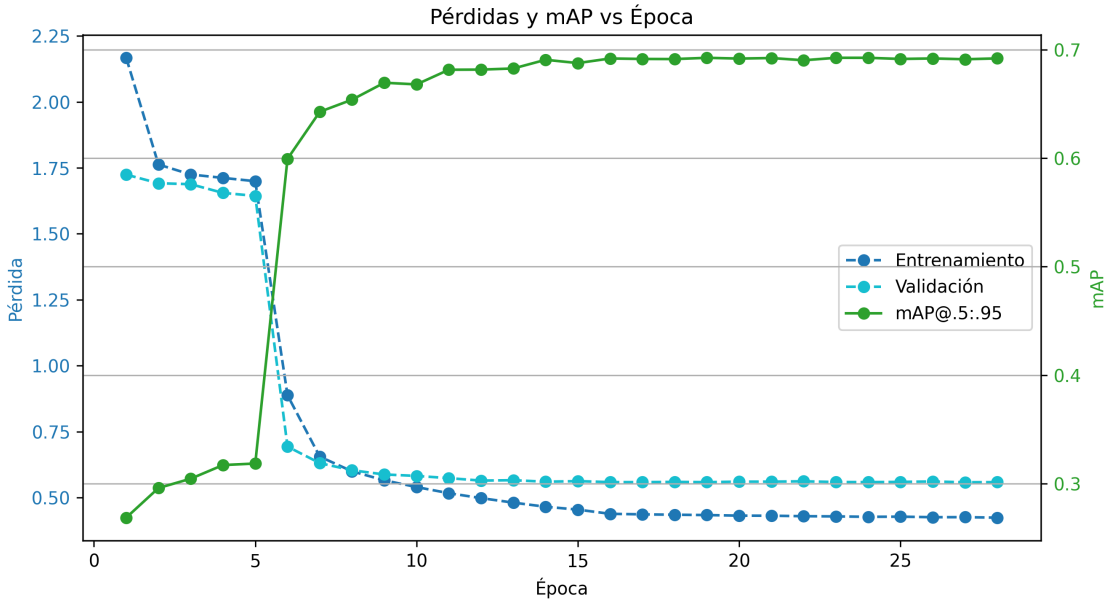


pérdida de entrenamiento disminuyó de 1.69 a 0.88 y la de validación de 1.64 a 0.69. En consecuencia, el mAP@0.5:0.95 paso de 0.31 a 0.59. Esto ocurrió gracias a que el backbone ResNet50 comenzó a ajustar sus filtros para detectar características más específicas de las hojas de tomate.

Durante las épocas 6-14, las pérdidas continuaron disminuyendo y el mAP siguió aumentando, pasando de 0.60 a 0.69. A partir de la época 15, las curvas de pérdida se aplanaron notablemente: la pérdida de entrenamiento siguió disminuyendo lentamente, mientras que la de validación se estabilizó. La pequeña brecha que se mantuvo entre ambas pérdidas indicó que el modelo no sufrió sobre ajuste. Finalmente, el mAP@0.5:0.95 alcanzó su valor máximo de 0.692 en la época

Figura 6

Curvas de entrenamiento y validación



23. Tras cinco épocas sin mejora, se activó la parada temprana, guardando los pesos de este mejor modelo.

Tabla 1

Resultados de la evaluación del modelo Mask-RCNN

Dataset	**Segmentación**		**Detección**	
	mAP @ IoU=0.5:0.95	AR @ maxDets=100	mAP @ IoU=0.50:0.95	AR@maxDets=100
Validación	0.692	0.808	0.656	0.781
Prueba	0.674	0.789	0.643	0.764

Nota. Esta tabla muestra los resultados del modelo en el dataset de prueba. mAP = Precisión media promedio; AR = Recuperación media; IoU = Intersección sobre Unión.

Una vez completado el entrenamiento, el modelo con los mejores pesos fue evaluado en el conjunto de prueba, que no fue utilizado durante el entrenamiento, para medir su capacidad de

generalización. En la tabla 1 se observa que en la tarea principal de segmentación de instancias, el modelo alcanzó un mAP @ IoU=0.50:0.95 de 0.67. El Average Recall para las máscaras fue de 0.78, indicando que el modelo fue capaz de identificar el 79 % de todas las hojas presentes en el conjunto de prueba. Los resultados para la detección de cajas delimitadoras (Bounding Box) fueron consistentes, con un mAP @ IoU=0.50:0.95 de 0.64. La pequeña diferencia entre el mAP obtenido en el conjunto de validación (0.69) y el conjunto de prueba (0.67) muestra la robustez del modelo ante datos desconocidos.

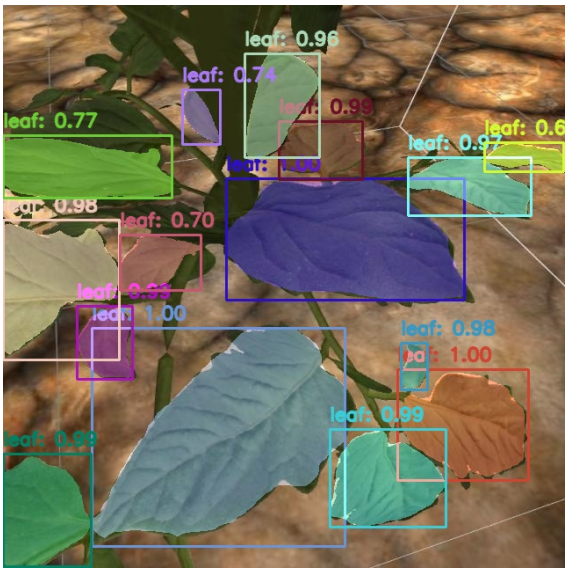
La figura 7 ofrece un análisis cualitativo del rendimiento del modelo en cuatro imágenes representativas del conjunto de prueba. En general, se observa que el modelo es robusto, generando predicciones con valores altos de confianza tanto en imágenes reales como en simuladas y manejando adecuadamente la variabilidad en formas, tamaños y fondos de las hojas.

Las imágenes segmentadas en la figura 7 muestran tanto aciertos como limitaciones. En la predicción (1), por ejemplo, aunque la detección es correcta, se aprecia que algunas de las máscaras no se ajustan con total fidelidad a la curvatura de la hoja. La predicción (2) ilustra la habilidad del modelo para segmentar hojas de distintos tamaños en una misma escena; a pesar del acierto general, el modelo comete dos errores: omite la segmentación de algunas hojas y fusiona dos hojas que se encuentran parcialmente superpuestas.

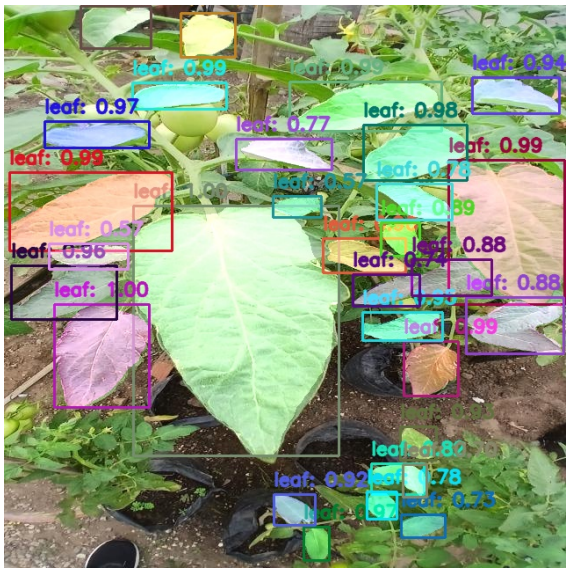
Para validar el flujo de trabajo completo del sistema de percepción, desde la adquisición de datos hasta la generación de la pose objetivo, se presenta un caso de estudio en la Figura 8. Esta figura ilustra la secuencia de transformaciones de datos que realiza el paquete object_point_cloud_processor. El proceso comienza con las imágenes RGB y de profundidad (Fi-

Figura 7

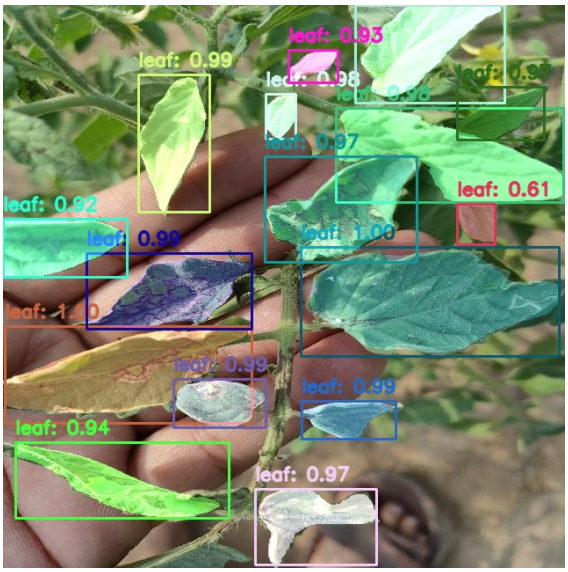
Hojas segmentadas con el modelo entrenado



(a) *Predicción 1*



(b) *Predicción 2*



(c) *Predicción 3*

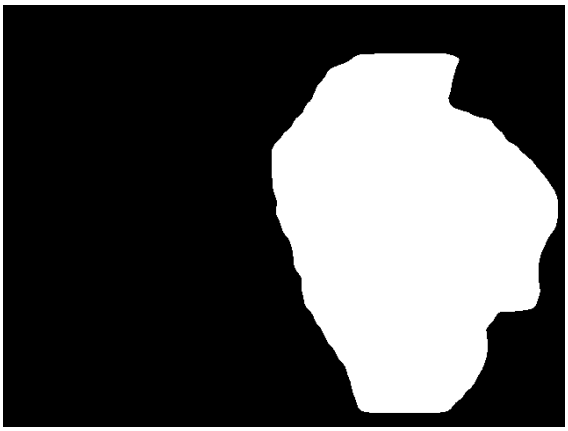
(d) *Predicción 4*

Figura 8

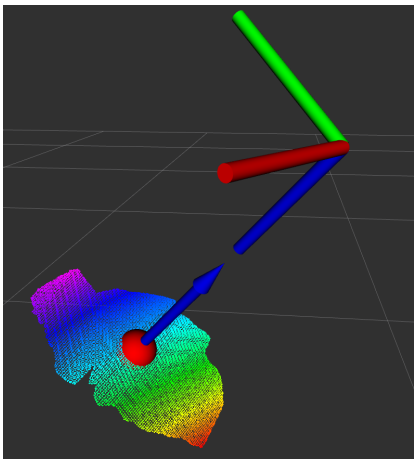
Secuencia completa del pipeline de percepción 3D

(a) *Imagen RGB*

(b) *Imagen de profundidad*



(c) *Máscara generada*



(d) *Nube de puntos y pose generada*

guras 8.a y 8.b), que son los datos proporcionados por la cámara simulada. La Figura 8.c muestra el resultado del segmentation_node, una máscara binaria que delimita el contorno de la hoja de interés, separándola del fondo y de otras partes de la planta.

Finalmente, la Figura 8.d presenta el resultado del pipeline. La nube de puntos, generada

por el cloud_processing_node, representa la geometría 3D de la hoja. Sobre esta nube se superpone la salida del normal_vector_extraction_node: una esfera que marca la posición del centroide, una flecha que indica la dirección del vector normal a la superficie y un sistema de coordenadas que representa la pose objetiva generada.

4.3 Validación del Sistema Integrado

Las pruebas se realizaron en un procesador Ryzen A12, se dividió la planta verticalmente en 3 zonas, zona baja (0-0.45 m), zona media (0.45-0.9 m) y zona alta (0.9-1.35 m), por cada zona se realizaron 8 pruebas. En cada una de las pruebas el brazo inicia en una configuración que le permite ver la mayor parte de la planta con el fin de crear el octomap de colisión que usara moveit2 para crear el plan de movimiento. Después el brazo se mueve a una pose generada aleatoriamente dentro de un rango de las articulaciones que asegura que la planta estará en el campo de visión de la cámara, desde allí se toma la fotografía y se inicia el pipeline que finaliza con la predicción del estado fitosanitario de la hoja. Los resultados de las 24 pruebas se muestran en la tabla 2.

Tabla 2

Resultados de prueba del sistema integrado

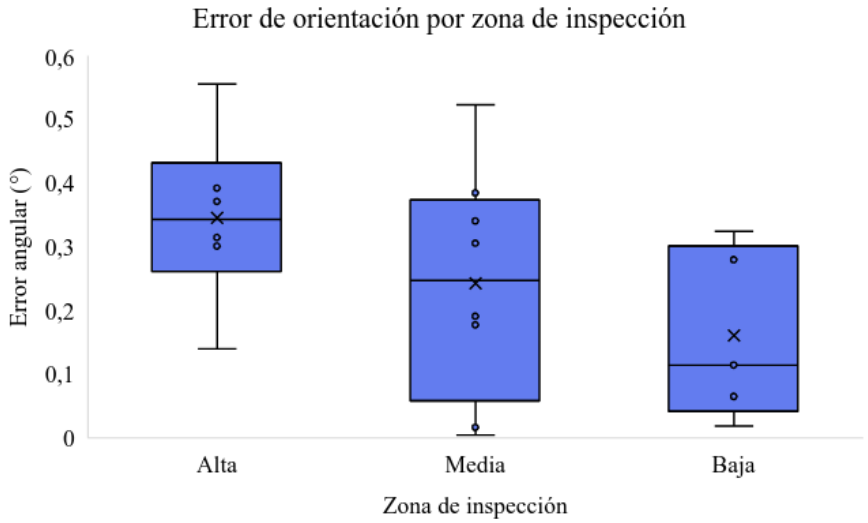
ID	Zona	Confianza	Distancia [m]	T. captura [s]	T. planificación [s]	T. movimiento [s]	E. orientación [°]	E. posición [m]	Éxito	T. total [s]
1	Alta	0,9897	0,5760	41,8569	15,7850	-	-	-	No	57,6419
2	Alta	0,9971	0,5524	45,9612	1,2637	18,3727	0,302	0,0025	Sí	83,1463
3	Alta	0,9962	0,5190	40,7411	1,4455	21,3171	0,393	0,00165	Sí	78,7725
4	Alta	0,9979	0,5299	48,7090	2,4150	2,4150	0,315	0,00066	Sí	84,4730
5	Alta	0,9949	0,3033	42,6553	14,3840	-	-	-	No	57,0393
6	Alta	0,9947	0,7885	47,5501	2,2096	21,2796	0,556	0,0009	Sí	79,5035
7	Alta	0,9961	0,5765	33,8992	2,9559	11,6669	0,140	0,00112	Sí	55,1456
8	Alta	0,9991	0,2223	53,3724	1,8186	15,0462	0,372	0,00079	Sí	85,3806
9	Media	0,9966	0,5467	46,1232	19,6589	-	-	-	No	65,7821
10	Media	0,9977	0,1467	55,4799	6,9840	25,0579	0,178	0,00086	Sí	122,5784
11	Media	0,9986	0,1876	47,9361	5,8266	15,8819	0,525	0,00638	Sí	108,7160
12	Media	0,9663	0,2253	42,1421	3,2569	37,2891	0,017	0,00045	Sí	121,5653
13	Media	0,9953	0,2342	39,1007	5,6556	36,2868	0,004	0,00012	Sí	117,4535
14	Media	0,9614	0,1649	40,3733	1,8859	23,8722	0,191	0,00112	Sí	87,7726
15	Media	0,9206	0,2040	37,1236	1,9248	14,7056	0,385	0,00044	Sí	83,1389
16	Media	0,9990	0,1994	46,5165	1,8719	15,0709	0,306	0,00036	Sí	81,4477
17	Baja	0,9993	0,3171	44,5690	2,4641	17,0472	0,341	0,0018	Sí	107,0324
18	Baja	0,9991	0,2779	52,0195	2,0278	21,8487	0,065	0,00019	Sí	121,3956
19	Baja	0,9965	0,3875	38,5144	1,9336	18,2937	0,324	0,0023	Sí	85,2390
20	Baja	0,9993	0,1883	47,4015	2,9557	23,3972	0,280	0,00012	Sí	100,8728
21	Baja	0,9976	0,2994	44,1817	1,7428	12,3297	0,115	0,00053	Sí	73,0769
22	Baja	0,9968	0,2796	35,2865	3,9759	16,7500	0,019	0,00065	Sí	86,2345
23	Baja	0,9563	0,2139	48,2059	16,9410	-	-	-	No	65,1469
24	Baja	0,9484	0,2865	37,2829	17,2517	-	-	-	No	54,5346

Nota. Esta tabla muestra los resultados de las pruebas de validación de todo el sistema.

El error entre la orientación calculada y la alcanzada fue pequeño, la diferencia máxima fue 0.556° y se obtuvo un promedio global de 0.254°. En la figura 9 se observa que la media del error disminuye a medida que la altura de captura es menor. El error de posicionamiento euclidiano también fue pequeño, con un promedio global de 1.3 mm.

Figura 9

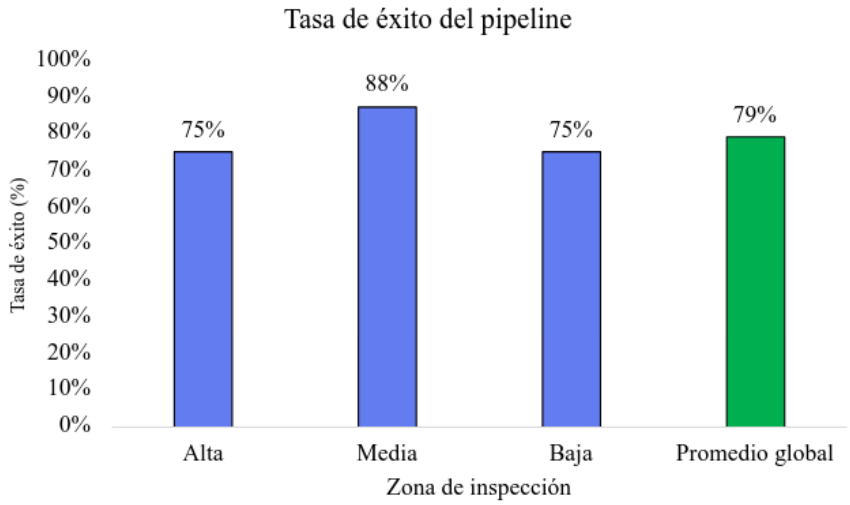
Error de orientación por zona de inspección



Como se observa en la tabla 2, el planificador no encontró una trayectoria válida hacia la hoja seleccionada en 5 ocasiones. En la zona alta, las 2 poses calculadas estaban por encima del espacio de trabajo del brazo. En la zona media, donde la planta es más voluminosa, la pose calculada estaba cerca de la base lo que provocaba que el brazo colisionara consigo mismo. En la zona baja, las 2 poses se encontraban dentro del contorno de la planta de forma que la presencia de otras hojas imposibilito la generación de una trayectoria libre de colisiones.

Figura 10

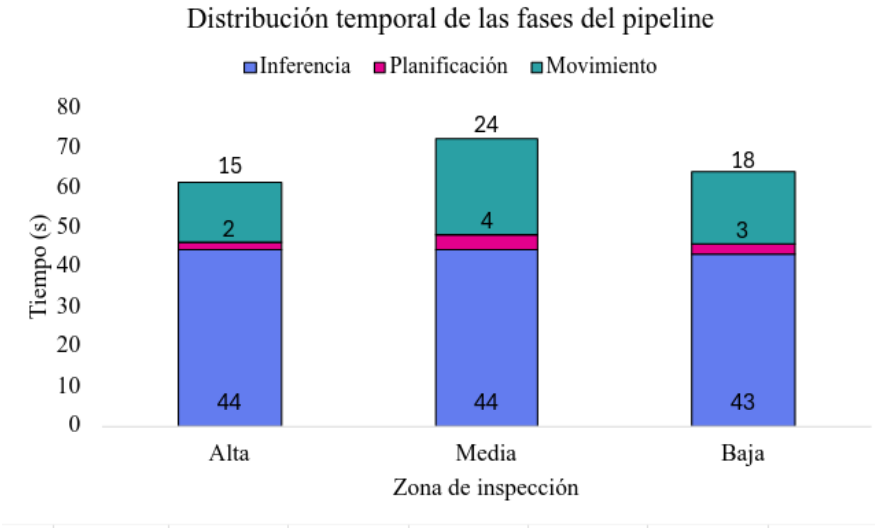
Tasa de éxito del pipeline por zona de inspección



Como se observa en la figura 10, el porcentaje de éxito en la zona alta es del 75 %, este rendimiento refleja las limitaciones de alcance máximo del brazo. En la zona media el robot opera lejos de sus límites articulares, permitiendo alcanzar un 88 % de éxito. En la zona baja, el suelo, la base y la planta disminuyen el espacio para maniobrar; como resultado, se obtiene una tasa de éxito del 75 %. El promedio de éxito global alcanzado durante las pruebas fue del 79 %.

Figura 11

Distribución temporal de las fases del pipeline



En la figura 11 se observa que la fase que toma más tiempo, con un promedio de 44[s] es la inferencia. De este tiempo, el modelo mask-rcnn usa 40[s] (90 %) en la segmentación, por lo tanto, este nodo es el cuello de botella del pipeline. Este tiempo es producto de la saturación de la CPU debido a la simulación y demás procesos. En la figura también se observa que los tiempos de planeación, ejecución y tiempo total son mayores en la zona media y baja, lo que evidencia una mayor complejidad geométrica.

5. Recomendaciones y Trabajos Futuros

A partir de los hallazgos experimentales y las limitaciones identificadas durante el desarrollo de esta investigación, se proponen las siguientes recomendaciones para mejorar la robustez y aplicabilidad del sistema en entornos agrícolas reales:

Se sugiere la incorporación de un grado de libertad prismático vertical en la base del mani-

pulador o el uso de una plataforma móvil con elevador, lo cual permitiría mitigar las limitaciones de alcanzabilidad observadas en los estratos superior e inferior de la planta. Asimismo, es fundamental mejorar la planta simulada en Gazebo incluyendo variabilidad en la forma, color y tamaño de las hojas.

Para hacer el sistema más robusto se puede añadir la inspección de la parte inferior de la hoja, ya que en esta zona se alojan también numerosas plagas y patologías.

Para optimizar el flujo de trabajo, se propone la implementación de un sistema de memoria que registre las coordenadas de las hojas ya inspeccionadas, para no procesar varias veces la misma. Además, se recomienda incluir una pose de aproximación antes de la pose final.

Finalmente, para despliegue o replicación de este trabajo se recomienda usar una GPU o arquitecturas distribuidas, esto reducirá los tiempos de inferencia de 40 segundos a milisegundos.

6. Conclusiones

- Se logró la integración del manipulador robótico UR5e y el modelo de la planta de tomate dentro del entorno de simulación Gazebo. Este entorno virtual demostró ser una herramienta eficaz para la validación segura del algoritmo de manipulación, permitiendo replicar las restricciones geométricas y de colisión propias de un cultivo real. Sin embargo, la uniformidad de las texturas representa una simplificación respecto variabilidad biológica real de una planta, lo cual debe considerarse para futuras reproducciones e implementaciones de este trabajo.

- Respecto al desarrollo de la estrategia de percepción, se validó el uso de datos RGB-D para

la extracción de características geométricas. La implementación de la red Mask R-CNN con un mAP@0.5:0.95 = 0.674 en el dataset de prueba, combinada con el análisis de nubes de puntos, permitió la extracción del vector normal de las hojas. Sin embargo, la ejecución del sistema sobre una CPU convencional presenta limitaciones de rendimiento significativas, registrando tiempos de inferencia elevados que restringen su operación en tiempo real bajo la configuración de hardware actual.

- Se configuró y validó el planificador RRTConnect (Rapidly-exploring Random Tree Connect) de la librería OMPL. Los resultados demostraron que este planificador es eficaz para resolver la cinemática inversa en espacios complejos de alta dimensionalidad, este permitió posicionar el brazo en la pose calculada con un error angular promedio de 0.25 grados, con tiempo promedio de planeación y ejecución de movimiento de 3.45[s] y 19.36[s] respectivamente. Además, a pesar de las limitaciones físicas de alcance y las colisiones en las Zonas Alta y Baja se alcanzó una taza de éxito global del 79 %.

- Finalmente, se construyó y validó el nodo para la clasificación de Tomato_Early_blight, Tomato_Late_blight, Tomato_healthy y Tomato_mosaic_virus utilizando una red ResNet50 preentrenada en el dataset PlantVillage. Las pruebas de integración confirmaron la correcta comunicación entre el pipeline de movimiento y el nodo de diagnóstico, validando el flujo completo de la operación.

Referencias Bibliográficas

Bae, B., Choi, Y., Jung, H., y Ahn, J. (2025). Tunnel lining segmentation from ground-penetrating radar images using advanced single- and two-stage object detection and segmentation models. *Computer-Aided Civil and Infrastructure Engineering*, *40*(22), 3580–3592. Descargado 2026-01-13, de https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.13528 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.13528) doi: 10.1111/mice.13528

Bao, Y., Shah, D. S., y Tang, L. (2018). 3D Perception-Based Collision-Free Robotic Leaf Probing for Automated Indoor Plant Phenotyping. *Transactions of the ASABE*, *61*(3), 859–872. Descargado 2024-03-23, de http://elibrary.asabe.org/abstract.asp?AID=49000&t=3&dabs=Y&redir=&redirType= doi: 10.13031/trans.12653

Choi, Y., Bae, B., Hee Han, T., y Ahn, J. (2024). Application of Mask R-CNN and YOLOv8 Algorithms for Concrete Crack Detection. *IEEE Access*, *12*, 165314–165321. Descargado 2026-01-13, de https://ieeexplore.ieee.org/document/10698754 doi: 10.1109/ACCESS.2024.3469951

Coleman, D. T., Sucan, I. A., Chitta, S., y Correll, N. (2014). Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study. Italy. Descargado 2025-08-05, de https://aisberg.unibg.it//handle/10446/87657 (Accepted: 2017-06-16T16:10:40Z) doi: 10.6092/JOSER_2014_05_01_p3

Craig, J. (2006). *Robótica: John J. Craig*. Pearson Educación. Descargado de https://books

.google.com.co/books?id=hRzOp_qdxG8C

farm. (2024, jul). *leaves_2 dataset* [Open Source Dataset]. https://universe.roboflow.com/farm-hu3nw/leaves_2. Roboflow. Descargado de https://universe.roboflow.com/farm-hu3nw/leaves_2 (visited on 2025-09-27)

Hartley, R., y Zisserman, A. (2004). *Multiple View Geometry in Computer Vision* (2.ª ed.). Cambridge: Cambridge University Press. Descargado 2025-07-18, de https://www.cambridge.org/core/books/multiple-view-geometry-in-computer-vision/0B6F289C78B2B23F596CAA76D3D43F7A doi: 10.1017/CBO9780511811685

He, K., Gkioxari, G., Dollár, P., y Girshick, R. (2018, enero). *Mask R-CNN*. arXiv. Descargado 2025-07-30, de http://arxiv.org/abs/1703.06870 (arXiv:1703.06870 [cs]) doi: 10.48550/arXiv.1703.06870

Jin, T., y Han, X. (2024, junio). Robotic arms in precision agriculture: A comprehensive review of the technologies, applications, challenges, and future prospects. *Computers and Electronics in Agriculture*, *221*, 108938. Descargado 2025-10-16, de https://www.sciencedirect.com/science/article/pii/S0168169924003296 doi: 10.1016/j.compag.2024.108938

Khan, S. U., Alsuhaibani, A., Alabduljabbar, A., Almarshad, F., Altherwy, Y. N., y Akram, T. (2025, mayo). A review on automated plant disease detection: motivation, limitations, challenges, and recent advancements for future research. *Journal of King Saud University Computer and Information Sciences*, *37*(3), 34. Descargado 2025-07-07, de https://doi.org/10.1007/s44443-025-00040-3 doi: 10.1007/s44443-025-00040-3

King, A. (2017, abril). Technology: The Future of Agriculture. *Nature*, *544*(7651), S21–S23. Descargado 2025-07-02, de https://www.nature.com/articles/544S21a (Publisher: Nature Publishing Group) doi: 10.1038/544S21a

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., ... Girshick, R. (2023, abril). *Segment Anything*. arXiv. Descargado 2025-11-08, de http://arxiv.org/abs/2304.02643 (arXiv:2304.02643 [cs]) doi: 10.48550/arXiv.2304.02643

Kuffner, J., y LaValle, S. (2000, abril). RRT-connect: An efficient approach to single-query path planning. En *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)* (Vol. 2, pp. 995–1001 vol.2). Descargado 2025-11-14, de https://ieeexplore.ieee.org/document/844730 (ISSN: 1050-4729) doi: 10.1109/ROBOT.2000.844730

LeCun, Y., Bengio, Y., y Hinton, G. (2015, mayo). Deep Learning. *Nature*, *521*, 436–44. doi: 10.1038/nature14539

Liu, S., y Liu, P. (2022, enero). Benchmarking and optimization of robot motion planning with motion planning pipeline. *The International Journal of Advanced Manufacturing Technology*, *118*(3), 949–961. Descargado 2025-11-13, de https://doi.org/10.1007/s00170-021-07985-5 doi: 10.1007/s00170-021-07985-5

Lozano-Perez. (1983, febrero). Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computers*, *C-32*(2), 108–120. Descargado 2025-08-05, de https://ieeexplore.ieee.org/document/1676196 doi: 10.1109/TC.1983.1676196

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., y Woodall, W. (2022, mayo). Ro-

bot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, *7*(66), eabm6074. Descargado 2022-06-02, de https://www.science.org/doi/10.1126/scirobotics.abm6074 doi: 10.1126/scirobotics.abm6074

Martin, J., Ansuategi, A., Maurtua, I., Gutierrez, A., Obregón, D., Casquero, O., y Marcos, M. (2021). A Generic ROS-Based Control Architecture for Pest Inspection and Treatment in Greenhouses Using a Mobile Manipulator. *IEEE Access*, *9*, 94981–94995. (Conference Name: IEEE Access) doi: 10.1109/ACCESS.2021.3093978

mastersthesis. (2024, oct). *tomato_village_fixed dataset* [Open Source Dataset]. https://universe.roboflow.com/mastersthesis-ciixz/tomato_village_fixed. Roboflow. Descargado de https://universe.roboflow.com/mastersthesis-ciixz/tomato_village_fixed (visited on 2025-09-27)

Oliveira, L. F. P., Moreira, A. P., y Silva, M. F. (2021, junio). Advances in Agriculture Robotics: A State-of-the-Art Review and Challenges Ahead. *Robotics*, *10*(2), 52. Descargado 2025-07-02, de https://www.mdpi.com/2218-6581/10/2/52 (Number: 2 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/robotics10020052

Orthey, A., Chamzas, C., y Kavraki, L. E. (2023, septiembre). *Sampling-Based Motion Planning: A Comparative Review*. arXiv. Descargado 2025-11-14, de http://arxiv.org/abs/2309.13119 (arXiv:2309.13119 [cs]) doi: 10.48550/arXiv.2309.13119

Ortiz, L., Gonçalves, L., y Cabrera, E. (2017). *A Generic Approach for Error Estimation of Depth Data from (Stereo and RGB-D) 3D Sensors*. doi: 10.20944/preprints201705.0170.v1

Ouyang, C., Hatsugai, E., y Shimizu, I. (2022, julio). A Novel Modular, Extendable Mobile

Robot for Image Data Collection Task in a Greenhouse. En *2022 International Conference on Advanced Robotics and Mechatronics (ICARM)* (pp. 556–561). Descargado 2025-07-09, de https://ieeexplore.ieee.org/abstract/document/9959384 doi: 10.1109/ICARM54641.2022.9959384

Phillips, J., Bedrossian, N., y Kavraki, L. (2004). Guided Expansive Spaces Trees: a search strategy for motion- and cost-constrained state spaces. En *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004* (pp. 3968–3973 Vol.4). New Orleans, LA, USA: IEEE. Descargado 2025-11-15, de http://ieeexplore.ieee.org/document/1308890/ doi: 10.1109/ROBOT.2004.1308890

PickNik Robotics. (2025). *MoveIt 2 Documentation.* Descargado 2025-08-05, de https://moveit.picknik.ai/humble/index.html

Polgár, A. (2025, julio). *Random crop field generator for Ignition Gazebo.* Descargado 2025-08-12, de https://github.com/azazdeaz/fields-ignition (original-date: 2020-10-30T16:10:46Z)

Redmon, J., Divvala, S., Girshick, R., y Farhadi, A. (2016, mayo). *You Only Look Once: Unified, Real-Time Object Detection.* arXiv. Descargado 2025-11-07, de http://arxiv.org/abs/1506.02640 (arXiv:1506.02640 [cs]) doi: 10.48550/arXiv.1506.02640

Sanchez-Ante, G., y Latombe, J.-C. (2002, junio). A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. En (pp. 403–417). doi: 10.1007/3-540-36460-9_27

Sapkota, R., Ahmed, D., y Karkee, M. (2024, septiembre). Comparing YOLOv8 and Mask R-

CNN for instance segmentation in complex orchard environments. *Artificial Intelligence in Agriculture*, *13*, 84–99. Descargado 2026-01-13, de https://www.sciencedirect.com/science/article/pii/S258972172400028X doi: 10.1016/j.aiia.2024.07.001

Schor, N., Bechar, A., Ignat, T., Dombrovsky, A., Elad, Y., y Berman, S. (2016, enero). Robotic Disease Detection in Greenhouses: Combined Detection of Powdery Mildew and Tomato Spotted Wilt Virus. *IEEE Robotics and Automation Letters*, *1*(1), 354–360. (Conference Name: IEEE Robotics and Automation Letters) doi: 10.1109/LRA.2016.2518214

Sucan, I. A., y Kavraki, L. E. (2012, febrero). A Sampling-Based Tree Planner for Systems With Complex Dynamics. *IEEE Transactions on Robotics*, *28*(1), 116–131. Descargado 2025-11-15, de https://ieeexplore.ieee.org/document/5958629/ doi: 10.1109/TRO.2011.2160466

Universal Robots A/S. (2025). *UR5e user manual* (Manual técnico). Descargado 2025-10-20, de https://www.universal-robots.com/manuals/EN/PDF/SW5_24/user-manual-UR5e-PDF_online/710-965-00_UR5e_User_Manual_en_Global.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2023, agosto). *Attention Is All You Need.* arXiv. Descargado 2026-01-11, de http://arxiv.org/abs/1706.03762 (arXiv:1706.03762 [cs]) doi: 10.48550/arXiv.1706.03762

Wang, X. (2016). Deep learning in object recognition, detection, and segmentation. *Found. Trends Signal Process.*, *8*, 217-382. doi: 10.1561/2000000071

Williams, D., Macfarlane, F., y Britten, A. (2023, mayo). *Leaf Only SAM: A Segment Anything Pipeline for Zero-Shot Automated Leaf Segmentation.* Descargado 2026-01-13, de https://

arxiv.org/abs/2305.09418v2

Workspace, N. (2022, apr). *3class dataset* [Open Source Dataset]. https://universe.roboflow.com/new-workspace-crmxx/3class-iduv1. Roboflow. Descargado de https://universe.roboflow.com/new-workspace-crmxx/3class-iduv1 (visited on 2025-09-27)

Yang, S., Liu, P., y Pears, N. (2023, agosto). Benchmarking of Robot Arm Motion Planning in Cluttered Environments. En *2023 28th International Conference on Automation and Computing (ICAC)* (pp. 1–6). Descargado 2025-11-13, de https://ieeexplore.ieee.org/document/10275283 doi: 10.1109/ICAC57885.2023.10275283

Yung, M. L., Murawska-Wlodarczyk, K., Babst-Kostecka, A., Maier, R. M., Merchant, N., y Ooi, A. (2025, marzo). A User-Friendly Machine Learning Pipeline for Automated Leaf Segmentation in Atriplex lentiformis. *Bioinformatics and Biology Insights*, *19*, 11779322251344033. Descargado 2026-01-13, de https://doi.org/10.1177/11779322251344033 (Publisher: SAGE Publications Ltd STM) doi: 10.1177/11779322251344033

Zhang, L., Cai, K., Sun, Z., Bing, Z., Wang, C., Figueredo, L., ... Knoll, A. (2025, marzo). Motion Planning for Robotics: A Review for Sampling-based Planners. *Biomimetic Intelligence and Robotics*, *5*(1), 100207. Descargado 2025-11-15, de http://arxiv.org/abs/2410.19414 (arXiv:2410.19414 [cs]) doi: 10.1016/j.birob.2024.100207

Apéndices

Apéndice A. Repositorio de GitHub

El código utilizado en este proyecto se encuentra disponible en este repositorio de GitHub https://github.com/felipe112709/robot-manipulator.

Apéndice B. Dataset y script de entrenamiento de Mask R-CNN

El dataset y el Notebook utilizados para el entrenamiento del modelo Mask R-CNN se encuentran disponibles en este repositorio de GitHub https://github.com/felipe112709/Tomato-Leaf-Segmentation-Dataset.