UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Desarrollo de un oso hormiguero artificial

Informe de Proyecto de Grado presentado por

Mauricio Berois, Lucía de Oliveira y Eduardo Gastelú

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Gonzalo Tejera
Guillermo Trinidad

Montevideo, 17 de febrero de 2025

Agradecimientos

En esta etapa final de nuestras carreras de grado queremos agradecer a todos aquellos que contribuyeron con la realización de este proyecto.

Quisiéramos agradecer especialmente a nuestros tutores Gonzalo Tejera y Guillermo Trinidad, por darnos la oportunidad de trabajar en este fascinante proyecto así como también guiarnos de forma constante a lo largo del proyecto.

Finalmente queremos agradecer a nuestras familias, ya que no hubiera sido posible alcanzar esta meta sin la ayuda y el apoyo que siempre nos brindan.

Resumen

El presente trabajo describe el desarrollo de un sistema autónomo que detecta y sigue hormigas utilizando un robot hexápodo equipado con una cámara y algoritmos de visión por computadora e inteligencia artificial. Como objetivo a largo plazo, esta solución busca contribuir a reducir la dependencia de pesticidas, ofreciendo una alternativa más amigable con el medio ambiente.

El desarrollo del robot incluyó múltiples iteraciones, mejorando la estabilidad y adaptabilidad para operar en terrenos complejos como pasto, tierra y superficies inclinadas. Se implementó un modelo de visión basado en YOLOv8, entrenado con un conjunto diverso de datos que incluye imágenes reales de caminos de hormigas capturadas en la Facultad de Ingeniería, datasets externos y datos sintéticos generados mediante simulaciones en Unreal Engine 5. La combinación de aprendizaje automático con algoritmos heurísticos permitió implementar la detección de hormigas y la toma de decisiones en tiempo real.

Las pruebas realizadas abarcaron entornos controlados de laboratorio, terrenos al aire libre y simulaciones. Los resultados evidenciaron que el sistema puede detectar y seguir hormigas en diversos escenarios, aunque los entornos más complejos, como el pasto largo, plantearon desafíos para el modelo de detección. A pesar de estas limitaciones, el sistema mostró un desempeño aceptable en términos de estabilidad, movilidad y precisión en la identificación de objetivos.

Este proyecto sienta las bases para futuras investigaciones y mejoras, incluyendo la incorporación de nuevos sensores, optimización del hardware y entrenamiento de modelos más avanzados. Los resultados obtenidos resaltan el potencial de esta solución para convertirse en una herramienta práctica en el control de plagas agrícolas, contribuyendo a la reducción de costos y riesgos ambientales asociados al uso de pesticidas.

Palabras clave: Robótica, Visión por computadora, Inteligencia artificial, Detección autónoma, Hormigas

Índice general

Capítulo 1

Introducción

En la última década, la inteligencia artificial (IA) y la robótica han experimentado avances revolucionarios que han transformado múltiples sectores, incluida la agricultura. La denominada "agricultura de precisión" Mokariya y Malam, 2020 ha emergido como una respuesta a los desafíos de productividad y sostenibilidad, permitiendo optimizar el uso de recursos, reducir el impacto ambiental y mejorar el manejo de plagas en cultivos críticos Meena y Saini, 2024. Dentro de este contexto, la aplicación de técnicas de IA y sistemas robóticos para el control de plagas cobra especial relevancia, ya que los métodos tradicionales basados en el uso intensivo de pesticidas no solo resultan costosos, sino que también generan efectos adversos en el ecosistema y pueden favorecer la aparición de plagas resistentes.

El presente proyecto se centra en el diseño, desarrollo y validación de un robot hexápodo que sigue caminos de hormigas. Las hormigas pueden representar una amenaza para la producción agrícola Angulo et al., 2022. Por el modo que se mueven las hormigas, realizando caminos, se aborda el problema desde un enfoque combinando hardware robótico, algoritmos de visión por computadora y técnicas de toma de decisión.

Estudios previos han abordado la detección y seguimiento de hormigas mediante técnicas como redes neuronales convolucionales (ResNet-50) Wu, Cao y Guo, 2020, (Mask R-CNN) Imirzian et al., 2019 y algoritmos de segmentación (Watershed) Sabattini et al., 2023

Por ejemplo, Wu, Cao y Guo, 2020 propuso un sistema basado en ResNet-50 y filtros de Kalman para seguimiento en tiempo real, logrando altos valores de MOTA y MOTP, mientras que Sabattini et al., 2023 combinó eliminación de fondo y Watershed para detectar hormigas en campo abierto, superando en precisión y eficiencia. Imirzian et al., 2019 utilizó Mask R-CNN para segmentar las hormigas y analizar su comportamiento en entornos no controlados.

En cuanto a la navegación, investigaciones como Zangrandi, Arrigoni y Braghin, 2020 demostraron la viabilidad de hexápodos con servomotores ajustables para transitar en terrenos irregulares, y el framework PUTN integró SLAM y planificación de trayectorias (PF-RRT, NMPC) para evaluar la transitabilidad de ambientes complejos. Estos trabajos ofrecen estrategias útiles para seguir rutas dinámicas en contextos agrícolas. Sin embargo, no se encontraron antecedentes que combinan la detección de hormigas con el movimiento autónomo del sistema, ya que la mayoría se centró en mantener la cámara fija. Esta carencia destaca la novedad y relevancia de integrar detección robusta con navegación adaptativa, asegurando que el robot no solo identifique las rutas de hormigas, sino que también se desplace eficientemente en entornos desafiantes.

Los métodos tradicionales de monitoreo y control de plagas suelen basarse en inspecciones manuales y aplicaciones generalizadas de pesticidas, lo cual no solo es ineficiente, sino que también puede generar impactos ambientales adversos y favorecer la resistencia de las plagas.

Por ello, se propone el desarrollo de soluciones tecnológicas que permitan una detección temprana, selectiva y en tiempo real de la actividad de insectos en áreas agrícolas. La presente tesis aborda este desafío mediante el diseño y la implementación de un robot hexápodo capaz de seguir sus caminos, con la idea de que el proyecto pueda ser utilizado como base para encontrar hormigueros o zonas con alta actividad de hormigas con dicho robot. El objetivo general de este trabajo es desarrollar y validar un robot hexápodo que logre el seguimiento de caminos de las hormigas.

Entre los objetivos específicos destacan:

- Diseñar la arquitectura mecánica y electrónica del robot hexápodo, asegurando estabilidad y versatilidad en terrenos agrícolas, integrando una cámara para recolección de datos en tiempo real.

- Desarrollar algoritmos reconocimiento visual mediante visión por computadora, tomando en cuenta el comportamiento de las hormigas.
- Evaluar la eficacia del sistema a través de pruebas de laboratorio y en campo.

La metodología adoptada en este proyecto se fundamenta en un enfoque interdisciplinario que combina diseño mecánico, electrónica, algoritmos de visión por computadora y pruebas experimentales:

- Diseño y prototipado: Se desarrolló la estructura hexápoda del robot, inspirándose en la anatomía de los insectos, que ofrece múltiples modos de locomoción para adaptarse a diferentes superficies.
- Desarrollo algorítmico: Se implementó un algoritmo de seguimiento de caminos que utiliza imágenes para reconocer al camino y decidir hacia donde moverse.
- Integración de sistemas: Se incorporaron cámaras, sensores y módulos de procesamiento de datos, permitiendo la comunicación entre el hardware y el software del robot.
- Validación experimental: Se realizaron pruebas en condiciones controladas y en escenarios reales, evaluando parámetros como precisión en la detección, velocidad de respuesta y capacidad para seguir rutas predefinidas.

Las pruebas permitieron evaluar integralmente el desempeño del sistema como un conjunto en condiciones representativas de escenarios reales. Estos resultados sugieren que la combinación de modelos de visión y robótica pueden ofrecer una herramienta eficaz para el seguimiento de hormigas, con el potencial de monitoreo de plagas en la agricultura para reducir el uso de pesticidas y mejorar la sostenibilidad de los cultivos.

Una demostración de la solución que se presentará se puede ver en el siguiente video.

El presente informe se organiza en los siguientes capítulos:

- Capítulo 1: Introducción. Presenta el contexto, los antecedentes, el planteamiento del problema, los objetivos, la metodología y la estructura del documento.
- Capítulo 2: Revisión de antecedentes - Marco teórico y revisión de la literatura. Se abordan los fundamentos de la IA en agricultura, la biomimética en robótica y estudios relacionados sobre navegación inspirada en hormigas.
- Capítulo 3: Metodología. Detalla el diseño, desarrollo e integración del robot hexápodo, así como los algoritmos implementados y los protocolos de prueba. Además de detallarse el uso de un simulador de nuestra realidad.
- Capítulo 4: Resultados experimentales. Se exponen y analizan los datos obtenidos en pruebas de laboratorio, del simulador y de campo.
- Capítulo 5: Conclusiones y trabajo a futuro. Se resumen los hallazgos principales, se concluye sobre la viabilidad del enfoque y se proponen futuras líneas de trabajo.

Capítulo 2

Revisión de antecedentes

En las etapas iniciales de este proyecto, se realizó una búsqueda exhaustiva en Google Scholar para encontrar artículos relevantes sobre la detección y el seguimiento de hormigas. Nuestro objetivo era explorar soluciones existentes que pudieran orientar nuestro enfoque. Además, buscamos artículos relacionados con la navegación en terrenos irregulares para abordar los desafíos de seguir a las hormigas o sus rastros en entornos exteriores. Asimismo, se llevó a cabo una búsqueda extensiva de datasets relacionados con hormigas en plataformas como Roboflow y otras similares. Se seleccionaron aquellos datasets que aportaban información nueva y útil para el desarrollo de nuestro proyecto.

En las siguientes secciones se detallan los hallazgos de la investigación y las referencias clave que guiaron nuestra investigación sobre la detección y el seguimiento de hormigas, así como la movilidad robótica en entornos exteriores.

El primer punto del proyecto se basa en la detección de las hormigas, para esto se fundamentó la investigación en varios artículos Wu, Cao y Guo, 2020, Sabattini et al., 2023, Imirzian et al., 2019, que proponen diversas soluciones a estos problemas. Además, se consideró la posibilidad de detectar el camino que dejan las hormigas al desplazarse y no las hormigas propiamente dichas. Por lo tanto, se investigó sobre seguimiento de caminos en entornos adversos e irregulares Jian et al., 2022. Esto marcó nuestra primera línea de investigación y a continuación se presentarán los resultados del relevamiento realizado.

Por otro lado, una vez detectadas las hormigas, o el camino que es dejado por ellas, es necesario poder seguir estas hormigas, para esto se relevaron artículos de movilidad de robots en exteriores Marković et al., 2014, Zangrandi, Arrigoni y Braghin, 2020, que proponen soluciones al problema de seguir objetivos en una plataforma robótica en movimiento.

Por último, se investigaron datasets de hormigas en exteriores en caso de que fuera necesario entrenar una inteligencia artificial capaz de detectarlas. Estos datasets son presentados en las secciones 2.2.3, 2.2.3 y 2.2.3

2.1. Marco teórico

En este apartado, se presentan y definen conceptos fundamentales que son clave para la comprensión de los temas que se abordarán en los siguientes capítulos. Estos conceptos brindan el marco teórico necesario para entender el contexto, los enfoques y las metodologías que se estudiarán y aplicarán a lo largo del desarrollo del proyecto.

2.1.1. Redes neuronales

Las redes neuronales son un tipo de modelo de inteligencia artificial inspirado en la estructura del cerebro humano. Estas redes consisten en capas de nodos interconectados, donde cada nodo procesa una pequeña parte de la información y transmite los resultados a las siguientes capas. Estas capas están organizadas de manera jerárquica: la información fluye desde las capas de entrada, a través de varias capas ocultas, hasta llegar a la capa de salida, donde se genera la predicción o el resultado final. En particular, las redes neuronales convolucionales (CNN) son muy eficaces para el procesamiento de imágenes, ya que están diseñadas para captar características espaciales como bordes, formas y texturas.

ResNet-50

ResNet-50 es una arquitectura de red neuronal convolucional profunda, compuesta por 50 capas. Fue diseñada para superar el problema de la degradación en redes profundas, donde agregar más capas puede reducir el rendimiento. ResNet introduce bloques residuales, que permiten que la red aprenda identidades, ayudando a que las capas más profundas mejoren su precisión sin los problemas de sobreajuste. ResNet-50 puede ser utilizado como la base de la arquitectura para la detección de hormigas, permitiendo extraer descriptores de apariencia que son esenciales para identificar y diferenciar las hormigas en las imágenes.

Mask R-CNN

Mask R-CNN es una red neuronal convolucional que se considera avanzada ya que no solo se utiliza para la detección y clasificación de objetos si no que también tiene una alta capacidad de segmentación de objetos en imágenes. Su característica principal es que genera una máscara que define la forma precisa de cada objeto. Esto le permite realizar no solo la detección tradicional, donde se delimita un objeto con un cuadro (bounding box), sino también segmentar el objeto a nivel de píxel.

El proceso de Mask R-CNN consta de dos fases: en la primera, identifica las áreas de la imagen donde pueden encontrarse objetos. En la segunda, refina estas áreas para predecir la clase del objeto y extraer una máscara que define su contorno exacto. Esta capacidad es crucial para aplicaciones que requieren un alto nivel de precisión, como el seguimiento detallado de objetos en entornos complejos.

RPN

Region Proposal Network es una red convolucional totalmente conectada que es utilizada por R-CNN para obtener propuestas de regiones donde pueden encontrarse los objetos de interés. Esta consiste de un clasificador que determina las probabilidades de que un objeto los objetos de interés se encuentren en esa región, y un regresor que devuelve las coordenadas de esa propuesta.

2.1.2. Watershed Based Algorithm

El algoritmo Watershed Based es un método de segmentación de imágenes comúnmente utilizado en visión por computadora para separar distintos objetos en una imagen. Se basa en la idea de ver la imagen como una topografía, donde las intensidades de píxeles representan alturas. El algoritmo inunda esta topografía comenzando desde los puntos más bajos y creando cuencas o *"watersheds"*. A medida que el algoritmo avanza, las cuencas se expanden hasta que colisionan, lo que permite definir los bordes entre objetos adyacentes.

2.1.3. Métodos de seguimiento y localización

Existen varios métodos que permiten realizar seguimiento y localización en entornos complejos, especialmente útiles en robótica y visión por computadora.

Filtro de Kalman

El Filtro de Kalman es un conjunto de ecuaciones matemáticas que proporciona una solución computacional eficiente (recursiva) del método de mínimos cuadrados. El filtro es muy potente en varios aspectos: admite estimaciones de estados pasados, presentes e incluso futuros, y puede hacerlo incluso cuando se desconoce la naturaleza precisa del sistema modelado.

Lucas-Kanade

El método de Lucas-Kanade se utiliza para la estimación de flujo óptico. Este método asume que el flujo es continuo en el vecindario del píxel considerado y resuelve las ecuaciones básicas de flujo óptico para todos los píxeles en ese vecindario utilizando el criterio de "mínimos cuadrados". Es un método eficaz para estimar el movimiento de objetos en secuencias de video.

SLAM

SLAM es el acrónimo de *Simultaneous Localization and Mapping*. Es un conjunto de algoritmos utilizados en robótica y realidad aumentada que tiene dos objetivos principales: localización, es decir, determinar la posición y orientación de un robot o dispositivo en un entorno desconocido, y mapeo, que implica, construir o actualizar un mapa de un entorno mientras se navega por él.

Filtro de Partículas

PF es el acrónimo de Particle Filter o filtro de partículas en español. Es una técnica de estimación bayesiana que utiliza un conjunto de muestras (partículas) para representar una distribución de probabilidad de un estado, generalmente utilizado en problemas de localización y estimación en robótica. El filtro de partículas estima la posición y orientación del robot basándose en la información de los sensores y en un mapa conocido del entorno.

2.1.4. Planificación de movimientos

RRT

RRT es el acrónimo de Rapidly-exploring Random Tree. Es un algoritmo de planificación de movimiento que construye de manera incremental un árbol de búsqueda. Se utiliza para encontrar trayectorias en espacios de alta dimensionalidad, y es especialmente útil cuando se trata de entornos con obstáculos. El algoritmo RRT selecciona aleatoriamente puntos en el espacio y trata de conectarlos de manera eficiente, asegurándose de no chocar con obstáculos.

PF-RRT

PF-RRT se refiere a una combinación de dos técnicas populares en robótica y planificación de trayectorias: el filtro de partículas (PF) y el árbol de búsqueda aleatoria rápida (RRT).

NMPC

El Control Predictivo no Lineal Basado en Modelos (NMPC) es una técnica avanzada de control de procesos utilizada para gestionar sistemas dinámicos complejos y no lineales. Esta metodología se basa en un modelo matemático no lineal del sistema, utilizándolo para predecir el comportamiento futuro y optimizar las acciones de control en tiempo real.

GPR

GPR es el acrónimo de "Gaussian Process Regression"(Regresión de Proceso Gaussiano), es un método de aprendizaje automático supervisado para realizar regresión, es decir, predecir valores continuos a partir de datos de entrada.

2.1.5. Métricas

El análisis de rendimiento de las herramientas utilizadas en los trabajos relacionados no solo depende de su capacidad para detectar y seguir objetos, sino también de cómo se comportan en escenarios reales, donde factores como los falsos positivos, los falsos negativos y la precisión en el seguimiento juegan un papel crucial. Para evaluar estas herramientas de manera objetiva y cuantificable, es necesario recurrir a un conjunto de métricas que nos permitan describir su efectividad y fiabilidad en diferentes contextos.

Las siguientes métricas proporcionan una forma de medir y comparar la exactitud, la precisión y el comportamiento de las herramientas evaluadas, tanto en tareas de detección como de seguimiento de objetos en secuencias de video. Al utilizar estas métricas, podemos identificar fortalezas y debilidades específicas en el desempeño de cada herramienta, permitiéndonos una evaluación detallada y una mejora continua en los métodos utilizados.

A continuación, se presentarán las métricas que serán utilizadas para describir el rendimiento de las diferentes herramientas evaluadas en los trabajos relacionados:

- FP (Falsos positivos o *False Positive*)[1]: El número total de objetos que son identificados incorrectamente como verdaderos o coincidentes, cuando en realidad no lo son.

[1] Berois, de Oliveira y Gastelú, 2024.

- FN (Falsos Negativos o *False Negative*)[2]: El número total de objetos que no coinciden de forma satisfactoria, es decir, que deberían haber sido identificados como verdaderos pero no lo fueron.
- TP (Verdadero Positivo o *True Positive*)[3]: Ejemplo en el que el modelo predice correctamente la clase positiva.
- TN (Verdadero Negativo o *True Negative*)[4]: Ejemplo en el que el modelo predice correctamente la clase negativa.
- Precisión o *Precision*[5]: La precisión es la proporción de todas las clasificaciones positivas del modelo que realmente son positivas.Matemáticamente, se define de la siguiente manera:

$$Precision = \frac{\text{Positivos correctamente clasificados}}{\text{Todas las clasificaciones positivas}} = \frac{TP}{TP + FP} \tag{2.1}$$

- Tasa de verdaderos positivos (TPR) o *Recall*[6]: La tasa de verdaderos positivos (TPR), o la proporción de todos los positivos reales que se clasificaron correctamente como positivos, también se conoce como recuperación. La recuperación se define matemáticamente de la siguiente manera:

$$Recall = \frac{\text{Positivos correctamente clasificados}}{\text{Todos los positivos reales}} = \frac{TP}{TP + FN} \tag{2.2}$$

- Exactitud o *Accuracy*[7]: La exactitud es la proporción de todas las clasificaciones correctas, ya sean positivas o negativas.Se define matemáticamente de la siguiente manera:

$$Accuracy = \frac{\text{Clasificaciones correctas}}{\text{Total de clasificaciones}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

- IDs (Identificadores): El número total de intercambios de identidad durante el proceso de seguimiento.
- FM (Fragmentos): El número total de incidentes en los que el resultado del seguimiento se interrumpe, dividiendo la trayectoria real en partes separadas.
- MOTA: Multi-object Tracking Accuracy. La suma ponderada del promedio de exactitud de seguimiento de todos los videos. La ecuación para calcular MOTA es

$$MOTA = \frac{1 - (FP + FN + IDS)}{NUM\_LABELED\_SAMPLES} \tag{2.4}$$

  donde NUM_LABELED_SAMPLES es el número de total de muestras etiquetadas.
- mMOTA: Es la media de múltiples valores de MOTA.
- MOTP: Multi-object Tracking Precision. La suma ponderada del promedio de precisión de seguimiento de todos los videos. La precisión de seguimiento mide la diferencia entre la intersección y la unión de los cuadros delimitadores (bounding boxes) etiquetados y los que marca el algoritmo.
- mMOTP: Es la media de múltiples valores de MOTP.
- FR: El número de fotogramas siendo seguidos por segundo.
- F2-Score: El F2-score es una medida que se utiliza para evaluar la precisión de un modelo de clasificación, especialmente en contextos donde los datos pueden estar desbalanceados. Es una variante del F-score, que es la media armónica de la precisión y la sensibilidad. Mientras que el F1-score trata la precisión y la sensibilidad de manera equitativa, el F2-score da más peso a la sensibilidad que a la precisión. Esto puede ser útil en situaciones donde la detección de verdaderos positivos es más importante que la minimización de falsos positivos.

[2]Berois, de Oliveira y Gastelú, 2024.
[3]Berois, de Oliveira y Gastelú, 2024.
[4]Berois, de Oliveira y Gastelú, 2024.
[5]Berois, de Oliveira y Gastelú, 2024.
[6]Berois, de Oliveira y Gastelú, 2024.
[7]Berois, de Oliveira y Gastelú, 2024.

El F-Score se calcula de la siguiente manera:

$$F_\beta = \left(1 + \beta^2\right) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{2.5}$$

El F2-Score se calcula sustituyendo β por 2

2.2. Trabajos relacionados

A continuación se presenta una recopilación de los trabajos encontrados sobre el tema. Todo el relevamiento realizado para el estado del arte se encuentra en el gitlab del proyecto Berois, de Oliveira y Gastelú, 2024, y en este documento se presenta un resumen de lo investigado.

2.2.1. Reconocimiento de hormigas

Un paso importante para nuestro proyecto es detectar las hormigas. Se realizó un relevamiento del estado del arte en relación a la detección de hormigas y caminos. Los artículos que serán presentados son los que mostraron mayor utilidad para el proyecto y fueron publicados recientemente.

En los tres papers presentados Wu, Cao y Guo, 2020, Sabattini et al., 2023 y Imirzian et al., 2019 se utilizaron técnicas para detección en imágenes. Estas imágenes contienen hormigas, algunas en entornos de laboratorio, sobre fondo blanco, y otras son imágenes de hormigas en el campo.

En dos de los papers se calcularon métricas asociadas al rendimiento de las soluciones propuestas y luego se realizó la comparación de las métricas de dos de los tres papers presentados, dado que uno de ellos no presentó las mismas métricas que los otros dos.

Seguimiento de colonias de hormigas utilizando aprendizaje profundo

En el artículo "Accurate detection and tracking of ants in indoor and outdoor environments" Wu, Cao y Guo, 2020 se desarrolló un enfoque basado en aprendizaje profundo para el seguimiento de colonias de hormigas. La etapa de detección se organizó en dos fases, utilizando la red neuronal convolucional *ResNet-50* como base y *Position Sensitive Score Maps* para codificar las regiones de interés.

Se presenta la arquitectura propuesta en la figura 2.1.

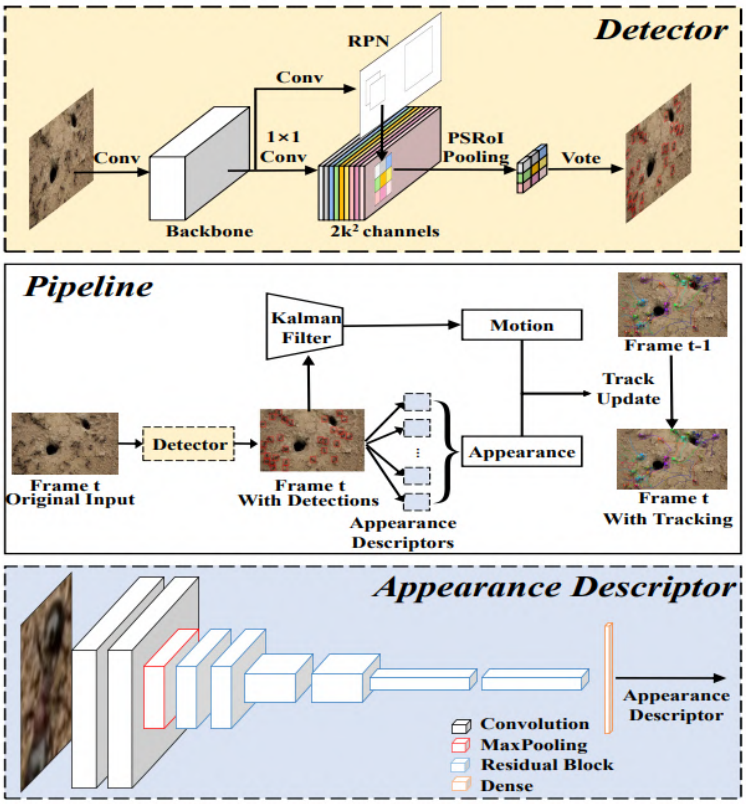


Figura 2.1: Arquitectura para detección y seguimiento tomado de Wu, Cao y Guo, 2020

Para la detección, el *backbone* corresponde a *ResNet-50* y las regiones de interés en las que podría haber un objeto son propuestas por RPN. Estas propuestas se refinan y se clasifican posteriormente mediante mecanismos de submuestreo y votación.

En esta arquitectura, el proceso comienza con un fotograma de entrada que pasa por una fase inicial de detección. Posteriormente, se implementa un Filtro de Kalman para analizar el movimiento de cada hormiga, lo que permite predecir el desplazamiento potencial de los objetos detectados. Para la caracterización de las hormigas, se emplea el modelo ResNet que genera descriptores de apariencia, los cuales son conjuntos de características extraídas de las imágenes o videos que permiten identificar y diferenciar a las hormigas entre sí. Estos descriptores se agrupan según sus similitudes, proceso que se ilustra en la última imagen. La fase final consiste en combinar esta información con la predicción de movimiento para actualizar el seguimiento.

El algoritmo construye secuencias de apariencias para las hormigas que luego son combinadas con la información de movilidad para realizar la tarea de seguimiento en tiempo real.

Para probar la eficacia de la solución propuesta se tomaron videos de hormigas, tanto para interiores como para exteriores, y se probó el algoritmo con esos videos así como también dos soluciones de detección de insectos ya existentes.

Al comparar el rendimiento del algoritmo desarrollado en esta investigación con dos programas de *software* de detección de insectos, idTracker y Ctrax, se concluyó que la velocidad de estas dos herramientas era entre seis y diez veces menor que la del algoritmo propuesto. Además, el algoritmo presentado ofrece mayores valores de precisión y exactitud en comparación con los otros dos. Estos resultados se pueden observar en la tabla 2.1 del paper previamente citado.

Method	FP ↓	FN ↓	IDs ↓	FM ↓	mMOTA ↑	mMOTP ↑	FR ↑
idTracker	881	8479	83	432	54	77.4	1.3
Ctrax	2832	5646	110	349	58.2	79.7	0.8
Solución presentada	**239**	**628**	**22**	**189**	**95.7**	**81.1**	**8.7**

Tabla 2.1: Comparación de *software* de detección de insectos
tomada de Wu, Cao y Guo, 2020

El código y los datasets utilizados fueron publicados para uso libre.

Visión por computadora de seguimiento de hormigas

El artículo "AntTracker: A low-cost and efficient computer vision approach to research leaf-cutter ants behavior" Sabattini et al., 2023 se enfocó en el desarrollo de un sistema de visión por computadora para el seguimiento de hormigas cortadoras de hojas en cultivos a través de videos.

Se presenta la arquitectura propuesta en la figura 2.2.

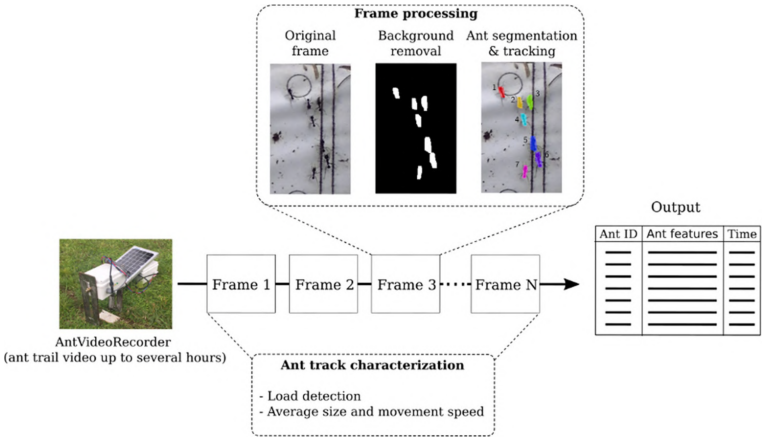


Figura 2.2: Pipeline propuesto para el proyecto, tomado de Sabattini et al., 2023

Para este proyecto se utilizó el hardware "AntVideoRecorder" para grabar la actividad de hormigas, permaneciendo fijo apuntando a un camino de hormigas. La grabación realizada por "AntVideoRecorder" es utilizada como entrada para realizar el procesamiento de los fotogramas del video. Este procesamiento es utilizado para remover el fondo de las imágenes recibidas y segmentar cada hormiga. Luego utilizando un conjunto de N fotogramas se realiza el seguimiento de cada una. Posteriormente, la información sobre cada una de ellas es utilizada para decidir si la hormiga está cargando con una hoja o no. Como resultado del procesamiento se obtiene un ID para cada hormiga junto con sus características y una marca de tiempo.

El sistema propuesto se basó en la detección de hormigas en el campo, contando cada hormiga con identificadores únicos. Se utilizó la herramienta "AntCounter", que cuenta el número de hormigas que pasan por el centro del fotograma de vídeo. Para identificar hormigas se realizó un proceso de etiquetado con el software "AntLabeler", usado para validar los resultados obtenidos por "AntTracker" en los videos de prueba, ambas herramientas pueden usarse en conjunto para etiquetar videos de multitudes de hormigas de manera asistida y semi-supervisada. Para la detección de hormigas se usó una mezcla de funciones gaussianas para eliminar el fondo y se utilizó el algoritmo Watershed Based Algorithm combinado con información de imágenes anteriores. Se llegó a esta combinación de algoritmos luego de probar varios métodos y que esta solución diera el mayor F2-Score y menor tiempo computacional.

Se compararon los valores de MOTA y MOTP para distintas variaciones del algoritmo elegido: utilizando el algoritmo de seguimiento por sí solo, luego utilizando también el área predicha, es decir, utilizando que una hormiga que carga una hoja ocupa un área mayor, luego agregando predicciones del Filtro de Kalman, y combinados. Como el movimiento de las hormigas no sigue necesariamente patrones lineales, decidir el siguiente movimiento de una hormiga es una tarea difícil. Aún así los estudios mostraron que el algoritmo de Kalman brinda diversas mejoras, como un ligero aumento en el valor del parámetro MOTP.

Por otro lado, tomando en cuenta la información extra del área que ocupan las hormigas, el aumento de MOTP es menor, esto se da tanto cuando se utiliza sólo esta información como cuando se combina con el Filtro de Kalman, por lo que parece ser más efectivo utilizar el Filtro de Kalman sin los valores del área. Los resultados demostraron ser efectivos y de bajo costo en la captura de información de la actividad de las hormigas cortadoras de hojas en el campo. Todo esto se puede ver en la figura 2.2.

El código fuente de la herramienta utilizada está disponible en github.

Algoritmo de asignación	MOTA ↑	MOTP ↑
Algoritmo base	0.960	7.020
Uso del área	**0.971**	7.048
Uso del filtro Kalman	0.970	**7.297**
Uso del área y filtro Kalman	**0.971**	7.046

Tabla 2.2: Comparación de algoritmos de seguimiento tomada de Sabattini et al., 2023

Identificación de hormigas y caminos troncales

El artículo "Automated tracking and analysis of ant trajectories shows variation in forager exploration" Imirzian et al., 2019 implementa una técnica de detección, tanto de las hormigas como sus caminos, con el fin de estudiar el comportamiento de estas en la naturaleza y su interacción con los riesgos presentes en la misma, como son los hongos en el caso de esta investigación, pero sería posible de aplicar a más riesgos, como insecticidas y otros insectos.

Los experimentos son llevados a cabo en el ambiente natural de estas hormigas, en la selva del sur de Brasil. Este entorno no controlado y complejo para la detección mediante imágenes aporta un valor significativo al estudio, ya que demuestra la capacidad del sistema para operar en condiciones reales y desafiantes. Para recabar datos se emplearon cámaras con filtros infrarrojos colocadas sobre los troncales próximos a hormigueros visibles desde la superficie. En cada camino se filmó un largo aproximado de 15cm, obteniéndose en total aproximadamente 79 horas de grabación, con 20,230,585 datos, de los cuales 8,412,477 se consideran de mayor utilidad. Se anotaron 20,666 imágenes con hormigas.

Para la detección de hormigas se entrenó un modelo de detección de objetos y segmentación utilizando Mask R-CNN. Con las hormigas identificadas en cada fotograma, se evalúan fotogramas sucesivos calculando el desplazamiento de cada hormiga para identificar así el camino utilizando métodos de seguimiento basado en transporte, donde las hormigas del fotograma anterior representan el proveedor, las del siguiente el receptor, y la carga está dada por el desplazamiento calculado. Si bien el objetivo del estudio difiere con el nuestro, estas técnicas utilizadas se ajustan a las necesidades de nuestro proyecto, donde sería viable utilizar redes neuronales y tener un sistema que detecte un camino para así poder seguirlo.

Otra parte relevante del estudio fue la identificación del comportamiento de cada hormiga, clasificándolo entre exploratorio y recolector, según el desplazamiento calculado, centrado en la rectitud del mismo. En principio este desarrollo no parecería estar relacionado con los objetivos del proyecto, pero puede llegar a ser un buen insumo a tratar que ayude a localizar caminos u hormigueros analizando y comprendiendo el comportamiento de las hormigas encontradas.

En la identificación de hormigas se reportó una precisión de 97.86 % en la especie de hormigas estudiada. Esta medida no es demasiado útil ya que solo se reporta la precisión y no se especifica la cantidad de entidades clasificadas erróneamente. Sobre la detección de caminos de hormigas se evaluó, según un observador humano, que en promedio el 78.70 % de los caminos no tenían errores en el conjunto de datos totales, mientras que este resultado mejoraba a 81.39 % en el conjunto de datos reducido que no reportaba casos difíciles, el cual mencionamos con anterioridad como el conjunto de mayor utilidad.

Los datasets utilizados son de público acceso.

Comparación entre AntTracker: A low-cost and efficient computer vision approach to research leaf-cutter ants behavior y Accurate detection and tracking of ants in indoor and outdoor environments

Algorithm	MOTA ↑	MOTP ↑
AntTracker	**97,1** %	**81,1** %
Accurate detection	95,7 %	72,97 %

Tabla 2.3: Comparación de los algoritmos presentados en las secciones anteriores

Como se puede observar en la tabla 2.3, la solución propuesta en AntTracker presenta un valor tanto de MOTA como de MOTP superior a la solución propuesta por Wu, Cao y Guo, 2020 que utiliza *ResNet-50* y *Position Sensitive Score Maps*.

Por otro lado no se tiene información de estas métricas para la solución propuesta por Imirzian et al.,

2019 cuya solución propone el uso de *Mask R-CNN* y el índice de rectitud (ratio entre el desplazamiento neto y el largo total del camino) como métrica de evaluación.

2.2.2. Seguimiento de objetos en terrenos irregulares

El segundo paso para nuestro proyecto es, una vez detectadas las hormigas, poder seguir el camino que generan. Es por esto que se realizó un relevamiento del estado del arte en relación a la navegación en terrenos irregulares y qué robots son aptos para esta tarea. Los artículos que serán presentados son los que mostraron mayor utilidad para el proyecto.

En dos de los papers se presentan técnicas para la navegación en entornos hostiles mientras que en el otro se presenta un prototipo de robot que se consideró podría ser útil en el entorno agrícola.

Detección y seguimiento de objetos en movimiento

El artículo "Moving object detection, tracking and following using an omnidirectional camera on a mobile robot" Marković et al., 2014 utiliza un robot equipado con una cámara omnidireccional para detectar objetos y luego realizar el seguimiento de los mismos.

La técnica utilizada consiste en calcular el patrón de movimiento aparente de los objetos dada una secuencia de imágenes que tenga las coordenadas del objeto. Esto se hace a través de la versión iterativa del algoritmo Lucas-Kanade, implementado en la librería OpenCV.

En la solución presentada, el algoritmo, dada las coordenadas actuales de un objeto, calcula dónde debería encontrarse a continuación el objeto en caso de que no se esté moviendo, dada la velocidad y dirección del robot. Luego se calculan las coordenadas obtenidas en la siguiente imagen, y en caso de existir una diferencia significativa entre las coordenadas calculadas anteriormente y las obtenidas en la nueva imagen, se asume que es porque el objeto se está desplazando, de lo contrario, se asume que el objeto está quieto.

Una vez que se detectaron los objetos que están en movimiento se calculan los valores que se le debería dar a la velocidad lineal y angular para que el robot se mueva en la dirección del objeto detectado.

Esta estrategia permite a un robot en movimiento seguir objetos que potencialmente también estén en movimiento de forma precisa.

No se cuenta con el código generado por esta investigación.

Robot y sistema de control para terrenos irregulares

En el artículo “Control of a Hexapod Robot Considering Terrain Interaction” Zangrandi, Arrigoni y Braghin, 2020 se presenta un sistema de control construido y probado sobre el robot hexápodo PhantomX AX Metal Hexapod Mark II de Trossen Robotics ilustrado en la figura 2.3. Este es capaz de desplazarse a través de terrenos irregulares, que presentan desniveles u obstáculos menores, de forma que todo o parte del robot puede transitar por encima de los obstáculos o zonas faltantes de terreno como pozos pequeños, y avanzar ajustándose a terrenos inclinados como elevaciones o pendientes, manteniendo la estabilidad del robot y una cierta altura constante del mismo sobre el suelo.

Estas habilidades logradas son útiles para atravesar los terrenos planteados en nuestro proyecto, a la vez de proporcionar una plataforma más favorable a las necesidades de nuestros sensores, como puede ser la cámara.

El sistema propuesto en este artículo utiliza doce servomotores AX-12A. Estos motores están programados para ejecutar caminatas preestablecidas y realizar correcciones dinámicas en cada extremidad cuando se interactúa con terrenos irregulares.

Figura 2.3: PhantomX AX Metal Hexapod Mark II

No se cuenta con el código generado por esta investigación

PUTN: Un Marco de Navegación para Terrenos Irregulares basado en Ajuste de Planos

En el artículo "PUTN: A Plane-fitting based Uneven Terrain Navigation Framework"Jian et al., 2022 se presenta un framework para la navegación en terrenos irregulares en tres pasos. Este framework fue diseñado para que un robot terrestre navegue de manera segura y efectiva en entornos con terreno irregular. Se propone un nuevo método de evaluación del terreno que integra diferentes indicadores, incluyendo pendiente, planitud y dispersión, para evaluar la transitabilidad del terreno.

Se presenta la arquitectura propuesta en la figura 2.4.

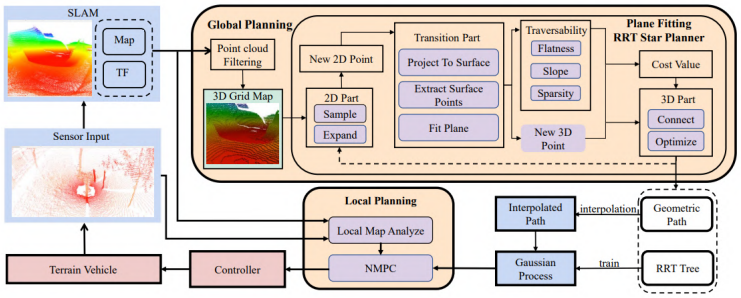


Figura 2.4: Vista del sistema del framework PUTN tomado de Jian et al., 2022

Este estudio presenta una técnica avanzada para la navegación de vehículos autónomos, un punto de interés para nuestro proyecto de seguimiento de hormigas. El vehículo utiliza sensores para generar una representación detallada de su entorno. Luego, a través de un método de localización y mapeo simultáneo conocido como SLAM por sus siglas en inglés, crea un mapa preciso del área. Este mapa se usa para planificar una ruta utilizando una combinación de técnicas que incluyen un algoritmo de exploración llamado PF-RRT y una regresión avanzada para refinar el camino basada en Regresión de Proceso Gaussiano (GPR). Finalmente, se usa un control predictivo no lineal basado en modelos (NMPC) para guiar el movimiento del vehículo de manera eficiente y segura.

La razón por la que este estudio es relevante para nuestro proyecto de seguimiento de hormigas radica en su enfoque para enfrentar entornos hostiles e irregulares, similares a los que encontramos al rastrear hormigas en un entorno agrícola. Las técnicas descritas en este framework, como la evaluación de terrenos mediante indicadores como pendiente, planitud y dispersión, pueden ser adaptadas para mejorar la precisión al seguir a las hormigas en condiciones difíciles. La capacidad del framework PUTN para analizar terrenos complejos y generar representaciones detalladas resulta particularmente útil para diseñar un sistema robusto que pueda operar en entornos agrícolas desafiantes, asegurando un seguimiento más confiable y eficiente de las rutas de las hormigas.

El código fuente está disponible de forma libre para consultar y utilizar.

2.2.3. Datasets

Con el objetivo de entrenar una inteligencia artificial capaz de detectar hormigas, se realizó un relevamiento del estado del arte en cuanto a datasets de hormigas.

Se listarán los datasets encontrados, de los cuales algunos fueron utilizados para el entrenamiento.

Dataset de trayectoria de movimientos de colonias de hormigas en exteriores e interiores con el fin de estudiar comportamiento en grupo

En el artículo "A dataset of ant colonies' motion trajectories in indoor and outdoor scenes to study clustering behavior" Wu et al., 2022 se recolectan diez videos en total, de colonias de tres tipos de especies, cinco videos realizados en exteriores y cinco videos realizados en interiores. Estos videos tienen etiquetadas a las hormigas con un software llamado VisualMarkData, en el artículo también se elabrora un programa para facilitar el trabajo de anotaciones en videos para dicha tarea. En total son 5343 fotogramas anotados con información y un numero identificador de cada hormiga, de un total de 712 hormigas y 114112 anotaciones, un ejemplo de esto se pude ver en la figura 2.5.

Kaggle ANT Detection Image Dataset

Kaggle, plataforma destinada a compartir datasets, cuenta con un dataset Shahane, 2021 que contiene 569 fotogramas de un video de hormigas en exteriores, moviéndose en un camino tomadas desde una vista aérea, etiquetadas. Las etiquetas estan realizadas en archivos de tipo detection y ground truth, también cuenta con imagenes similares en un entorno cerrado y de fondo blanco.

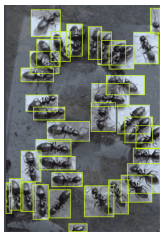
Datasets en Roboflow

En Roboflow contamos con varios datasets, conteniendo imagenes de hormigas por sí solas, y también en conjuntos, formando un camino.

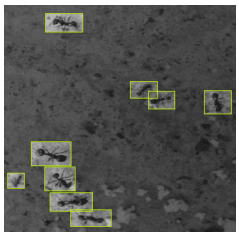
El usuario "Djay de Gier" cuenta con varios Datasets de hormigas en RoboFlow, mencionamos algunos:

- 14210 imágenes etiquetadas de hormigas en varios contextos de Gier, 2023c
- 9997 imágenes etiquetadas de hormigas realizado para un clasificador de hormiga. de Gier, 2023d
- 8494 imágenes etiquetadas de hormigas, de cerca, de lejos, solas y en grupo de Gier, 2023b
- 4127 imágenes etiquetadas de hormigas. de Gier, 2023a
- 1559 imágenes etiquetadas de hormigas, tomadas muy de cerca. de Gier, 2023f
- 844 imágenes etiquetadas de hormigas, identificando por especie de Gier, 2023h
- 868 imágenes de hormigas, solas y en grupo. de Gier, 2023g
- 841 imágenes de hormigas, solas y en grupo. de Gier, 2023e
- 1480 imágenes etiquetadas de hormigas. Usuario 'Proyecto Hormiga" en RoboFlow, 2023
- 1149 imágenes etiquetadas de hormigas Usuario 'intellaNest" en RoboFlow, 2023
- 1018 imágenes etiquetadas de hormigas. Usuario 'TFG" en RoboFlow, 2023
- 997 imágenes etiquetadas de hormigas. Usuario 'AntsNet.org" en RoboFlow, 2021
- 330 imágenes etiquetadas de hormigas. Usuario 'ants" en RoboFlow, 2022
- 295 imágenes etiquetadas de hormigas. Usuario 'Personal" en RoboFlow, 2022

No consideramos entrenar al modelo para que pueda detectar diferentes tipos de especies, aunque algunos datasets hicieran dicha distinción por especies, es razonable para el contexto de nuestro problema detectar hormigas, independientemente de la especie a la que pertenezcan, así que se les dará las imágenes de hormigas de todo tipo de especie identificadas solamente como hormigas, para que sepa detectar qué es una hormiga en general. En la figura 2.5 se presentan algunas imagenes de estos datasets, mostrando ejemplos del contenido que ofrecen.

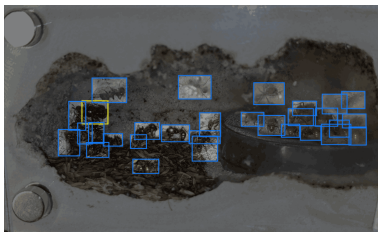


(a) Imagen del dataset Usuario ‘ants” en RoboFlow, 2022

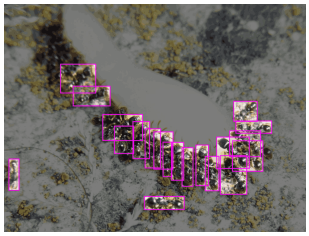


(b) Imagen del dataset Usuario ‘TFG” en RoboFlow, 2023

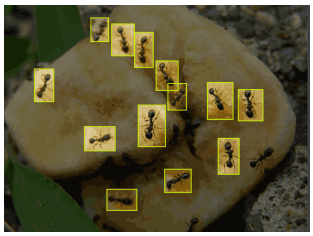
(c) Imagen del dataset de Gier, 2023c



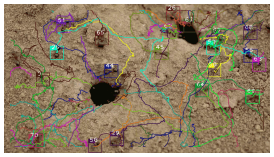
(d) Imagen del dataset Usuario ‘AntsNet.org” en RoboFlow, 2021



(e) Imagen del dataset Usuario ‘intellaNest” en RoboFlow, 2023



(f) Imagen del dataset Usuario ‘Personal” en RoboFlow, 2022



(g) Imagen del dataset Wu et al., 2022

Figura 2.5: Imágenes de varios datasets de hormigas.

Capítulo 3

Desarrollo

En este capítulo se detalla el proceso completo de diseño, implementación y pruebas llevadas a cabo para desarrollar el sistema de seguimiento autónomo de hormigas. Este sistema combina hardware, software y modelos de inteligencia artificial para resolver un problema complejo: detectar y seguir caminos de hormigas en entornos desafiantes.

Cada una de las secciones de este capítulo aborda un aspecto clave del desarrollo, permitiendo entender cómo se combinaron diferentes tecnologías y metodologías para resolver los desafíos planteados por este proyecto.

3.1. Robot hexápodo

Para este proyecto, se decidió utilizar un robot hexápodo como en Zangrandi, Arrigoni y Braghin, 2020 debido a su capacidad para adaptarse a entornos exigentes, como terrenos irregulares, desniveles, distintas alturas de pasto y superficies con tierra, entre otros obstáculos típicos de los hábitats naturales de las hormigas.

3.1.1. Hardware

Se utiliza una Odroid N2+ sobre el Robot, que controla los movimientos de los motores y la entrada de video. Esta posee un procesador ARM de 4 nucleos Cortex A73 de 2.4Ghz, y 2 nucleos Cortex A53 de 2Ghz, y 4GB de RAM DDR4 de 1320Mhz Amazon, 2024.

3.1.2. Estructura general

Primera versión

Teniendo en cuenta los materiales disponibles en el laboratorio de robótica de la Facultad de Ingeniería, se decidió utilizar el kit de robótica Bioloid para construir el robot King Spider Robotis, s.f. Esta primer versión del robot se puede ver en la figura 3.1:

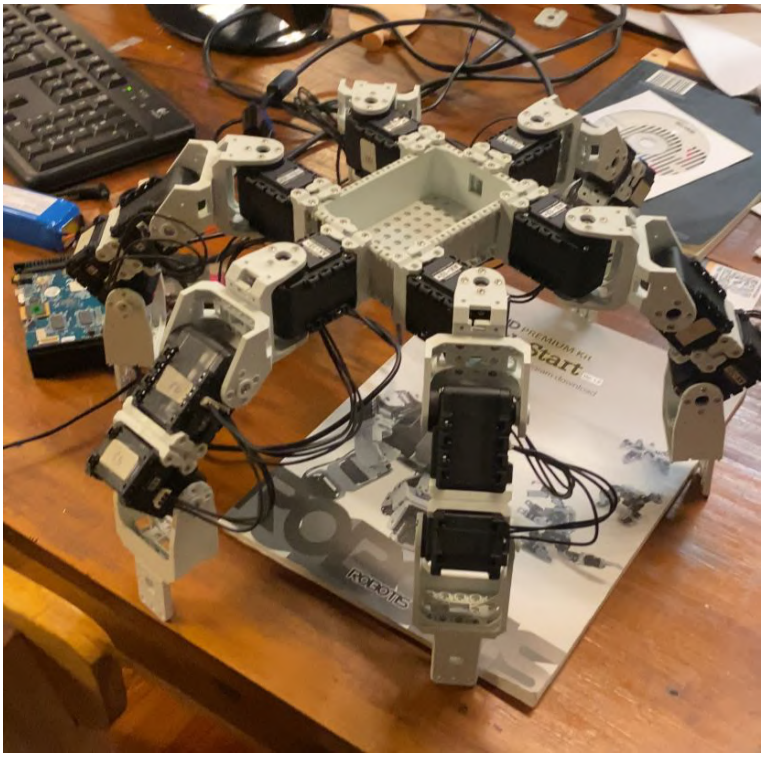


Figura 3.1: Primer versión del robot

Para comprobar que este robot era viable para el proyecto, se realizaron diversas pruebas. La primera de ellas consistió en dejar al robot andar hacia adelante por un minuto en diferentes ambientes. Los resultados de esta primera prueba se pueden observar en la tabla 3.1, esta tabla es un promedio de los cinco experimentos realizados en cada ambiente. El desplazamiento lateral representa cuánto se desvió el robot de su trayectoria prevista rectilínea. La desviación angular se calcula como el ángulo formado entre la dirección de avance deseada (línea recta hacia adelante) y la trayectoria real recorrida por el robot, proporcionando una medida en grados de cuánto se apartó de su curso original.

Ambiente	**Desplazamiento frontal ↑**	**Desplazamiento lateral ↓**	**Desviación ↓**
Alfombra	188 cm	3 cm	1°
Cemento	188 cm	4 cm	1°

Tabla 3.1: Experimento para testear robot inicial

Durante la realización del experimento, tanto en la alfombra como en el cemento, se observó que el robot presentaba una baja estabilidad. En muchas ocasiones se volcó, impidiendo que continúe el experimento.

Se identificaron las patas delgadas del robot como el primer punto de falla. Al ser finas y tener poco contacto con la superficie causaron que el robot pierda la estabilidad con frecuencia y se caiga. Por ello, se decidió cambiar las patas del robot, y basándose en los componentes del kit que se utilizó para construir el robot original, se identificaron tres nuevas opciones de posibles bases para las patas. Las cuatro opciones posibles se pueden observar en las figuras 3.2a, 3.2b, 3.2c, 3.2d.

Para cada configuración de patas, se realizó el mismo experimento en un ambiente controlado (alfombra) con el objetivo de observar si con alguna de las nuevas configuraciones presentaba mejoras en cuanto a la estabilidad del robot. Los resultados de estos experimentos se pueden observar en la siguiente tabla 3.2. Se incluyeron en el promedio las corridas que terminaron por fallas del robot, contabilizando un total de cinco corridas.

(a) Opción 1 (b) Opción 2 (c) Opción 3 (d) Opción 4

Figura 3.2: Posibles patas para el robot

Ambiente	**Desplazamiento frontal ↑**	**Desplazamiento lateral ↓**	**Desviación ↓**
Opción 1	128 cm	73 cm	30°
Opción 2	66 cm	83 cm	52°
Opción 3	115 cm	23 cm	11°
Opción 4	137 cm	0 cm	0°

Tabla 3.2: Experimento para testear posibles patas del robot

Segunda versión

Teniendo en cuenta el avance y la poca desviación que se consiguió con la configuración número cuatro se eligió esta opción para continuar con el robot. En la figura 3.3 se puede observar el robot con las patas seleccionadas. Esta configuración representa la segunda versión del robot construido.

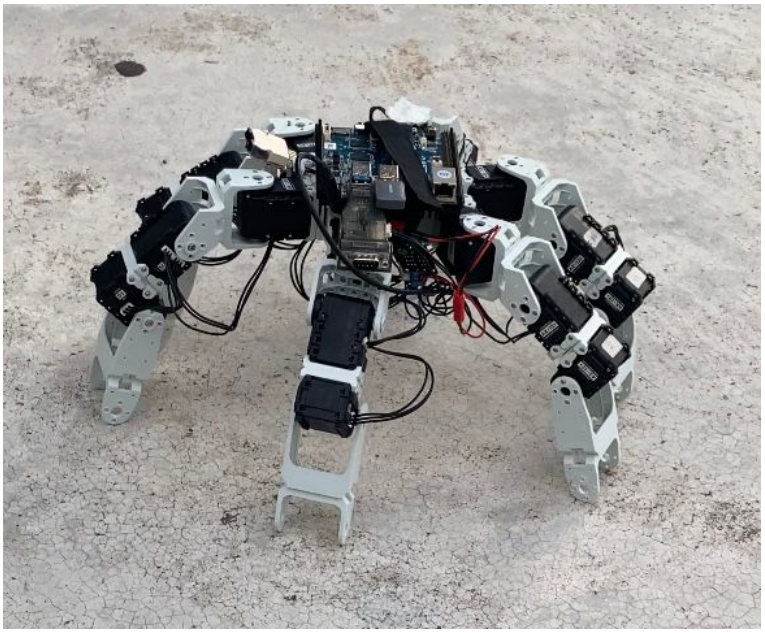


Figura 3.3: Segunda versión del robot

Una vez seleccionadas las patas del robot se procedió a realizar las pruebas en los ambientes: alfombra, cemento y pasto corto. El objetivo de este experimento es verificar si las patas elegidas eran óptimas para todos los ambientes de prueba. En la tabla 3.3 se observan los resultados.

Este experimento demostró que el robot no era capaz de avanzar en el pasto sobre el que se hicieron las pruebas. Tras un análisis, se llegó a la conclusión de que el mayor problema era la inclinación. Las pruebas en cemento, realizadas en bajada, mostraron que el robot se desplazaba con cierta eficacia. Sin embargo, en el pasto, donde el terreno era inclinado en subida, el robot se caía hacia atrás, impidiendo cualquier avance, y por eso su fila no tiene valores en la tabla 3.3. Para ver si era problema de ir en

Ambiente	Desplazamiento frontal ↑	Desplazamiento lateral ↓	Desviación ↓
Alfombra	137 cm	0 cm	0°
Cemento	144 cm	41 cm	16°
Pasto corto	–	–	–

Tabla 3.3: Experimento para testear posibles patas del robot

pasto, o de ir en terreno inclinado en subida, realizamos pruebas en el cemento, pero esta vez en sentido contrario, es decir, con inclinación hacia arriba. Los resultados mostraron que el robot también se caía hacia atrás en este entorno, indicando que el robot no tenía la estabilidad suficiente como para afrontar caminos que tuvieran subidas. Visto esto, se empezó a pensar un diseño distinto para el robot.

Tercera versión

- **Nuevo Modelo** Debido a las dificultades de movilidad constatadas en las pruebas presentadas anteriormente, se concluyó que la altura del robot, que concentra su peso en la parte más alta, combinada con el pequeño tamaño del área formada entre las patas que forman el apoyo en cada paso, causaban que con inclinaciones muy leves el robot perdiera el equilibrio y cayera, dándose vuelta de tal forma que todas las patas pierden el apoyo y no fuera capaz de recuperarse.

  Con el fin de solucionar esa problemática, se decidió enfocarse en modificar dos aspectos claves del diseño del robot: el tamaño del área de equilibrio formada por los puntos de apoyo y la altura del robot, puesto que eran los puntos más factibles a ser modificados para lograr mejorar la estabilidad del robot significativamente.

  Expandir el área de equilibrio puede lograrse aumentando el alcance de las patas e incrementando el tamaño del cuerpo del robot. Esta última opción era la de mayor interés y había sido considerada anteriormente ya que también permitiría una nueva disposición de patas donde el robot tendría tres patas en cada lateral. Esta configuración proporcionaría a las patas frontales una posición adecuada y espacio suficiente para realizar movimientos al avanzar, evitando así su comportamiento hasta el momento, en el cuál las patas frontales eran arrastradas por las demás al avanzar. Debido a esto se procedió a diseñar una nueva base para el robot.

- **Impresión** Se decidió realizar una impresión 3D para construir una nueva pieza central por la facilidad y libertad de acción que nos daba esta herramienta para transformar la estuctura de la base actual. Esto permitió construir un cuerpo mejor adaptado a nuestras necesidades con una mayor confiabilidad y precisión que trabajando manualmente con otros materiales disponibles como modelos en madera. Para mantener la compatibilidad con las patas del robot se tomó como referencia las uniones laterales de la pieza central del kit de Robotis Bioloid que utilizábamos y se adaptaron las medidas de largo y ancho tanto como las dimensiones de la impresora nos permitían, pasando de 10cm x 7cm a 20.5cm x 14cm, duplicando así el tamaño del cuerpo central.

  El aumento del tamaño del robot trajo leves mejoras al equilibrio del robot, pero el mayor aumento de rendimiento se encontró en la capacidad de reposicionar las patas a la nueva configuración. Bajo esta nueva configuración, las bases para las patas elegidas anteriormente con el fin de minimizar la fricción en el arrastre perdieron su utilidad - ya que el robot dejó de arrastrar las patas como parte de su comportamiento - y en cambio presentaban la desventaja de una falta de tracción que producía que el robot patinara o resbalara en ocasiones.

  En la figura 3.4 se puede observar la nueva configuración descrita, la cual será la versión final del robot que utilizaremos en el resto del proyecto.

- **Base de las patas** Se volvieron a realizar experimentos con las diferentes patas, agregando un nuevo tipo de pata a la prueba, la cual se encuentra en la figura 3.5. Esta incorporación fue necesaria debido a que las configuraciones de patas utilizadas hasta el momento presentaban fallas catastróficas, haciendo que el robot se resbalara y terminara cayendo en todas las pruebas realizadas. Los datos completos del experimento pueden encontrarse en el anexo A.

  Esta vez el experimento fue llevado a cabo en el pasto corto, que representa mejor las condiciones donde se esperaría trabaje el robot en la práctica. La opción mostrada en la figura 3.5 obtuvo los mejores resultados ya que, como fue comentado, fue la única disposición que no sufrió alguna falla

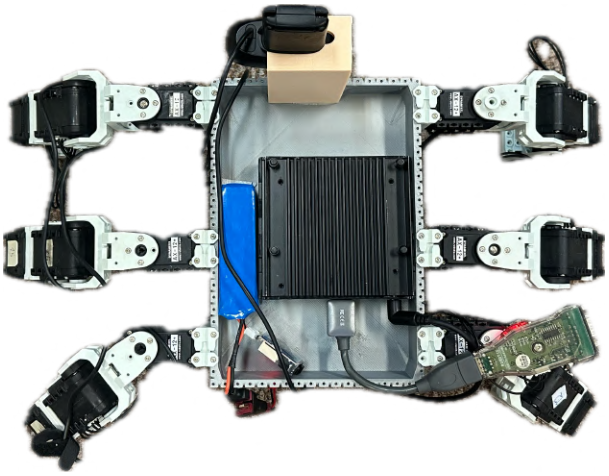


Figura 3.4: Modelo final del robot

catastrófica - en los demás casos el robot resbalaba y se daba vuelta - por lo que se decide continuar el proyecto con ellas.

Al haberse definido el nuevo cuerpo, la disposición de las patas y su apoyo, se decidió realizar nuevamente pruebas en distintos entornos para evaluar el rendimiento del robot tras los cambios y poder constatar en las pruebas las mejoras esperadas en comparación con las pruebas anteriores. Se mantuvo el tiempo total de la prueba, con corridas de 60 segundos. A continuación se presenta el promedio de las mediciones realizadas, para cada uno de los ambientes se realizaron 5 pruebas.

Ambiente	**Desplazamiento frontal** ↑ (cm)	**Desplazamiento lateral** ↓ (cm)	**Desviación** ↓ (°)
Alfombra	85 ± 3.00	11.4 ± 2.70	7.68 ± 2.10
Pasto Corto	80.4 ± 1.34	10.2 ± 2.39	7.24 ± 1.77
Arena	113.6 ± 3.58	27.8 ± 8.35	13.77 ± 4.20
Pasto Largo	64.8 ± 21.76	6.6 ± 8.65	4.89 ± 4.88

Tabla 3.4: Tabla de promedios y desviaciones estándar de los resultados del experimento

Figura 3.5: Opción 5 para patas del robot

3.1.3. Movimientos del robot

Para el control de los movimientos del robot, se utiliza la biblioteca ax12_control Aakieu, 2021 para Python que permite el manejo de motores AX12. Optamos por dicha librería ya que es una biblioteca minimalista, que contiene las funcionalidades justas y necesarias para controlar el movimiento del robot, además de ya estar integrada con Python, el mismo lenguaje en el que utilizamos YOLO y el sistema heurístico de control del robot.

Dentro de esta biblioteca se utilizaron las siguientes funciones:

- Ax12.connect(): Función utilizada para setear el baudrate y el puerto a utilizar
- Ax12(*id_motor*): Función utilizada para generar una conexión con el motor correspondiente al *id_motor* que se pasó por parámetro
- Ax12(*id_motor*).set_goal_postion(*position*): Función utilizada para que el motor correspondiente al *id_motor* que se pasó por parámetro gire hasta la posición *position* especificada.
- Ax12(*id_motor*).set_moving_speed(*speed*): Función utilizada para indicarle al motor correspondiente al *id_motor* que utilice la velocidad *speed* a la cual moverse.

El robot hexápodo desarrollado utiliza un patrón de locomoción denominado "trípode alterno". Este método se basa en la coordinación de las seis patas del robot, divididas en dos grupos de tres patas cada uno. Estos grupos se ven ilustrados en la figura 3.6, las patas azules representan el trípode A mientras que las amarillas representan el trípode B. Se presenta un video ilustrativo del movimiento del robot.

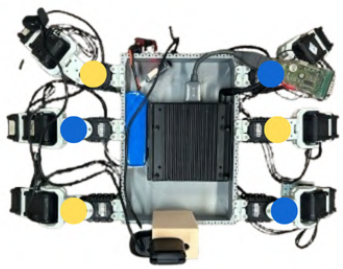


Figura 3.6: Subdivisión en trípodes

Para realizar los movimientos se decidió la altura a la cual se elevará cada pata y cuánto rotara. En el caso de la altura se realizaron pruebas en pasto y se estableció una altura en la cual las patas no se engancharan perjudicando la estabilidad del robot. Para el caso de la rotación, se eligió la mayor rotación posible para cada pata, evitando cualquier interferencia con las patas adyacentes.

A continuación, se describe el ciclo completo del movimiento hacia adelante:

1. Fase de elevación y rotación del primer trípode: En esta fase, tres patas del robot se elevan simultáneamente del suelo. Estas tres patas son luego rotadas hacia adelante en un movimiento sincronizado, preparando su nueva posición de apoyo.
2. Fase de descenso del primer trípode: Una vez que las patas del trípode A han sido movidas a su nueva posición, descienden al suelo para proporcionar soporte y propulsión.
3. Fase de elevación del segundo trípode: Las otras tres patas del robot se elevan del suelo. El trípode A vuelve a su posición inicial, generando que el robot se mueva hacia adelante.
4. Fase de rotación del segundo trípode: Las patas del trípode B también son rotadas hacia adelante en un movimiento coordinado, preparándose para su próximo punto de apoyo.
5. Fase de Descenso del Segundo Trípode: Finalmente, las patas del trípode B descienden al suelo y proporcionan soporte al robot, completando así el ciclo de movimiento.

A continuación se presenta una máquina de estados que muestra los pasos del robot en cada instancia. Cada vez que el robot realiza un movimiento se espera un determinado tiempo y luego se procede a realizar el siguiente movimiento.

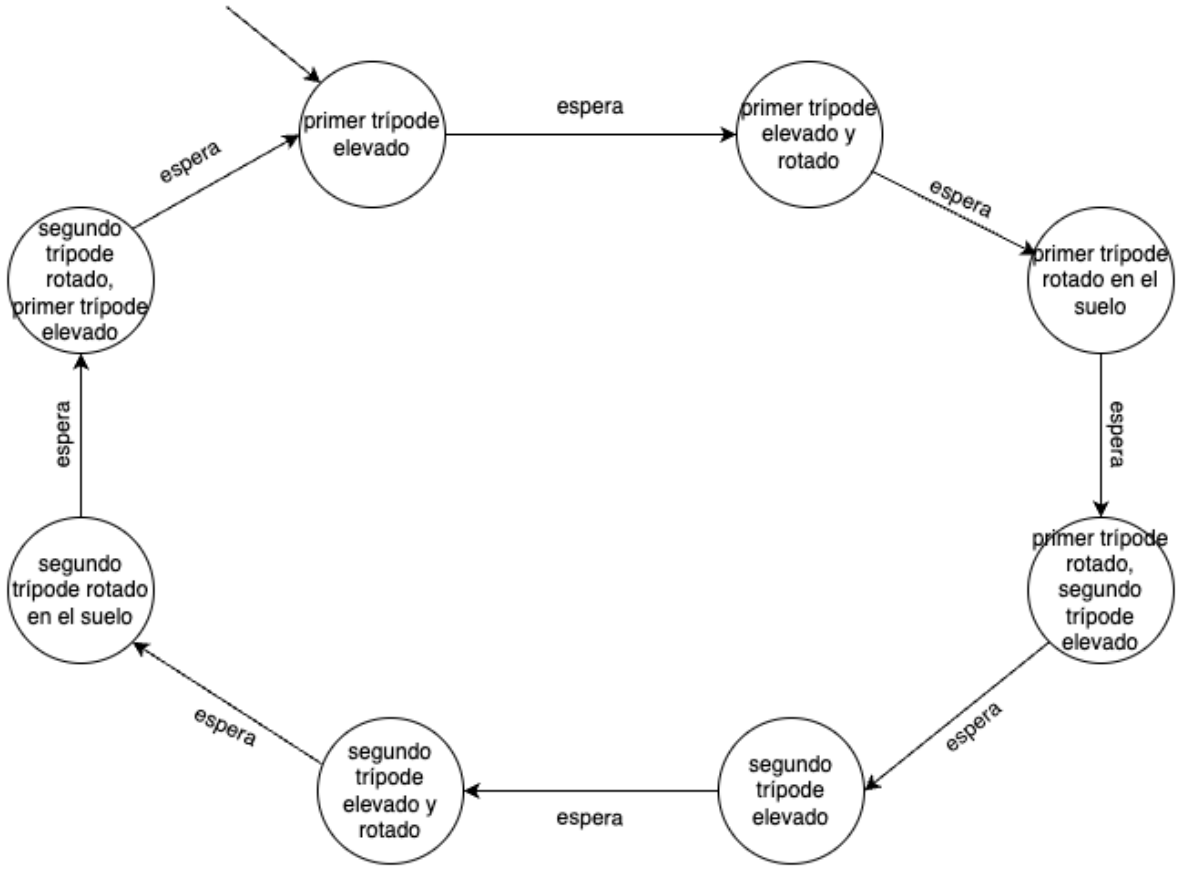


Figura 3.7: Máquina de estado del movimiento del robot

El movimiento alternado de los trípodes garantiza que siempre haya al menos tres patas en contacto con el suelo, proporcionando estabilidad durante todo el proceso de locomoción.

A continuación se describe el giro del robot, ambos giros son análogos, intercambiando la dirección de giro.

1. Fase de elevación y rotación del primer trípode: En esta fase el trípode A se eleva del suelo. Las tres patas son luego rotadas en sentido horario/antihorario en un movimiento sincronizado, preparando su nueva posición de apoyo.
2. Fase de descenso del primer trípode: Una vez que las patas del trípode A han sido movidas a su nueva posición, descienden al suelo para proporcionar soporte y propulsión.
3. Fase de elevación del segundo trípode: El trípode B se eleva del suelo. El trípode A vuelve a su posición inicial, generando que el robot comience el giro hacia la dirección estipulada.
4. Fase de rotación del segundo trípode: Las patas del trípode B también son rotadas en el mismo sentido que las del trípode A en un movimiento coordinado, preparándose para su próximo punto de apoyo.
5. Fase de Descenso del Segundo Trípode: Finalmente, las patas del trípode B descienden al suelo y proporcionan soporte al robot, completando así el ciclo de movimiento.

Entre cada una de las fases especificadas, para los tres movimientos del robot, se utilizó la sentencia de python *time.sleep*, permitiendo que cada trípode finalizara su movimiento antes de que el movimiento del otro trípode comenzara, garantizando el equilibrio, ya que siempre había tres patas que estaban apoyadas en el suelo.

Pruebas de Velocidad y Tiempo de Espera

Con la estructura del robot ya establecida y probada, el siguiente paso consistió en evaluar y mejorar la movilidad del robot a nivel de comportamiento. Se investigaron otros tipos de caminatas, entre ellas el movimiento que levanta siempre dos patas del robot, y mantiene el resto como apoyo, pero se resolvió mantener la caminata de tres patas a la vez, en la cual siempre hay tres patas de apoyo, debido a que producía movimientos más rápidos con una menor complejidad y se adaptaba mejor a la disposición de las patas. Los parámetros restantes del comportamiento a evaluar eran la velocidad de los motores, y el tiempo de espera entre cada movimiento, el cual es utilizado para sincronizar los pasos del robot, debido a que las instrucciones a los motores no son bloqueantes. Para ello, se realizó el siguiente experimento.

Se utilizó únicamente el entorno del laboratorio, con piso de alfombra, y se siguieron los parámetros de medición ya establecidos, distancia recorrida en x e y en 60 segundos y la desviación calculada a partir de ambas medidas.

Los motores utilizados reciben una velocidad entre 0 y 1024, cuando la velocidad se setea en 1024 se ejerce el máximo torque, que es 1.5 N.m Robotis, 2024. Luego, dependiendo del valor que se le de, el torque que se aplica; si se setea la velocidad en 512 el torque aplicado será de 0,75 N.m.

Para elegir los valores a probar se comenzó con los valores actuales y en cada iteración se aumentó el valor de la velocidad - a tramos de 100 debido a que una cantidad menor no producía una diferencia relevante - o se disminuyó el valor de la espera, probando así las distintas combinaciones entre los mismos. El valor máximo de velocidad evaluado fue 400, ya que a partir de ese valor se presentaba un notorio descontrol en el movimiento y dirección del robot, aumentando la desviación y por ende disminuyendo la distancia recorrida. El valor mínimo de tiempo de espera fue de 0.3 segundos, puesto que al probar valores menores ocurrían fallas en los motores como desconexiones, o las patas se acercaban demasiado unas a otras pudiendo engancharse entre sí. Por ello, se volvieron a los valores de 0.3 segundos y superiores que permitían un funcionamiento normal y seguro del robot.

Como se puede observar en la tabla 3.5, donde se presentan los promedios obtenidos a partir de cinco corridas realizadas con la misma configuración de velocidad y tiempo de espera, los valores óptimos para estos parámetros son una velocidad de 300 y un tiempo de espera de 0.3 segundos. Con estos valores se obtuvieron distancias recorridas superiores al doble de las distancias recorridas iniciales, y alcanzando desviaciones de cero y dos grados, con una desviación en promedio de 4.4 grados. Esta desviación es inferior a la desviación promedio de 7.6 grados obtenida con los parámetros iniciales de 200 de velocidad y tiempo de espera de 0.5 segundos.

Velocidad	**Time.sleep**	**Desplazamiento frontal** ↑ (cm)	**Desplazamiento lateral** ↓ (cm)	**Desviación** ↓ (°)
200	0.5 s	85 ± 3.00	11.4 ± 2.70 cm	7.68 ± 2.10
200	0.3 s	178.8 ± 23.62	84.0 ± 20.76	24.97 ± 4.09
300	0.5 s	108.2 ± 3.70	56.6 ± 7.70	27.61 ± 3.96
300	**0.3 s**	**198.6 ± 3.13**	**15.4 ± 14.17**	**4.44 ± 4.12**
400	0.3 s	154.6 ± 3.44	99.4 ± 2.19	32.74 ± 0.87

Tabla 3.5: Tabla de promedios y desviaciones estándar de los resultados de A.2

3.1.4. Cámara

En este apartado se detalla el proceso de selección y configuración de la cámara utilizada en el sistema de seguimiento de hormigas. Dado que la calidad de las imágenes y la precisión en la detección son factores críticos para el correcto funcionamiento del algoritmo, se evaluaron varios modelos de cámaras disponibles en el laboratorio. Además, se diseñó un soporte específico para asegurar la correcta posición y estabilidad de la cámara durante las pruebas. A continuación, se describen los criterios utilizados para la elección de la cámara, así como los detalles del diseño y montaje del soporte.

Elección de cámara

Se evaluaron tres modelos de cámaras disponibles que se pueden observar en la figura 3.8

Durante las pruebas iniciales, se descartó el modelo Xiaomi CMSXJ22A por realizar efecto de “ojo de pez”, distorsionando los bordes de la imagen, lo que dificulta la toma de decisiones de movimiento. La precisión de la detección se beneficia significativamente cuando la cámara minimiza las deformaciones

(a) Xiaomi CMSXJ22A

(b) Logitech C210

(c) Logitech C270

Figura 3.8: Modelos de cámaras disponibles en el laboratorio de la facultad

de las líneas rectas. El modelo Logitech C210 fue descartado por su baja resolución en comparación con las otras cámaras. Por último, la cámara Logitech C270 fue la seleccionada por su mínima distorsión y su mejor resolución dentro de las tres. En la figura 3.12 se presentan fotografías capturadas desde cada cámara, todas enfocadas en el mismo punto.

Figura 3.9: Punto de vista del robot con la cámara Xiaomi CMSXJ22A

Figura 3.10: Punto de vista del robot con la cámara Logitech C210

Figura 3.11: Punto de vista del robot con la cámara Logitech C270

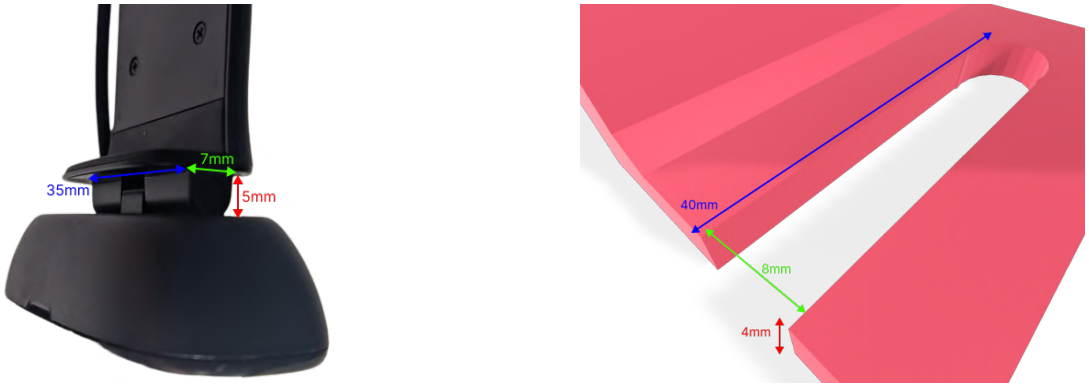
Figura 3.12: Punto de vista del robot con diferentes cámaras

Posición y altura de cámara

Para asegurar la estabilidad y la correcta posición de la cámara, fue necesario diseñar e imprimir un soporte específico.

Una vez seleccionada la cámara, se determinó la altura para montarla de manera que enfocara adecuadamente el suelo sin perder de vista a las hormigas. Después de realizar varias pruebas, se decidió una altura de 16 cm desde el suelo. Al probar alturas menores, la cámara perdía el enfoque, y al probar alturas mayores, no ganábamos un mejor enfoque ni claridad, solamente perdíamos detalle de lo que se encontraba en el suelo al alejarnos del objetivo.

La cámara cuenta con un pivote de 5mm de altura, 35mm de ancho y 7mm de fondo como vemos en la figura 3.13a. El diseño del soporte se realizó considerando dichas dimensiones, con un hueco en la cara horizontal de 4mm de altura, 40mm de ancho y 8mm de fondo para contener al pivote, el cual vemos en la figura 3.13b.



(a) Dimensiones del pivote de la cámara seleccionada

(b) Dimensiones del espacio que contiene a la cámara

Figura 3.13: Dimensiones del pivote y del soporte de la cámara

Las dimensiones de la cara que sostiene la cámara son de 50 mm x 33 mm, mientras que la cara conectada a la base del robot mide 99 mm x 50 mm, esta medida fue tomada para que al sumarle la altura del robot al estar de pie, resulte en tener la cámara a 16cm de distancia al suelo.

El soporte se fijó a la base del robot utilizando adhesivo. A continuación se ve cómo quedó el robot luego de incluir la cámara en la figura 3.14.

Figura 3.14: Robot con la cámara integrada

3.2. Unreal Engine 5

Unreal Engine 5 (Engine, 2024) fue seleccionado como plataforma para el desarrollo de un simulador en el cual realizar caminos de hormigas por su capacidad para crear entornos virtuales y a su integración robusta con herramientas de inteligencia artificial orientadas a personajes. Estas herramientas son esenciales para simular el movimiento de grandes multitudes de personajes que siguen rutas predefinidas como lo es un camino de hormigas. Además, al ser uno de los motores gráficos más utilizados en la actualidad, Unreal Engine 5 cuenta con una extensa documentación y una activa comunidad de soporte, lo que facilita la resolución de problemas y la implementación de soluciones. Una de las principales ventajas de Unreal Engine 5 es su alto grado de personalización, lo que nos permitió extender la plataforma con funciones personalizadas, tanto mediante su propio sistema de scripting como a través de C++. Esto fue fundamental para necesidades específicas del proyecto, como la generación de archivos de bounding boxes para el dataset, o la implementación de comandos por API REST que permiten controlar la simulación desde agentes externos.

La plataforma seleccionada es eficaz para simular un contexto en el cual utilizar la lógica de toma de decisiones de movimiento del robot, sin necesidad de hardware físico ni disponer de hormigas reales en un entorno controlado. Permite evaluar y ajustar los algoritmos de seguimiento y detección en un entorno virtual antes de implementarlos en el robot real.

El simulador genera automáticamente conjuntos de datos de imágenes etiquetadas, lo que resulta útil para entrenar modelos de visión.

Desarrollamos una simulación del entorno del robot que permite al usuario controlar una cámara en primera persona. El usuario puede mover la cámara hacia adelante, así como girarla en sentido horario y antihorario en torno al eje vertical, lo que simula los movimientos generales de nuestro robot. Además, el simulador permite crear y editar caminos de hormigas en el suelo, trazándolos en el editor y especificando la cantidad de hormigas que recorrerán dichos caminos.

3.2.1. Hardware

Para el desarrollo y ejecución de la simulación en Unreal Engine se utilizó una PC de escritorio equipada con un procesador Intel Pentium G4560 @ 3.50GHz, una tarjeta de video AMD RX 580 8GB y memoria RAM DDR4 8GB.

3.2.2. Obtención del modelo y animación de hormiga

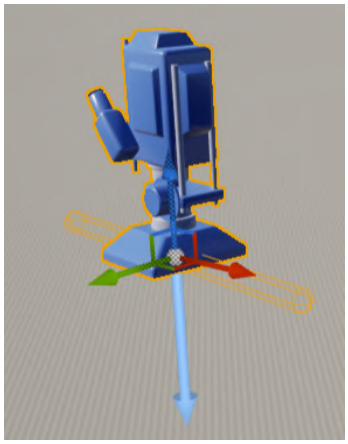
Se utilizó un modelo 3D y animación de caminata de una hormiga tomado de SketchFab MAXDESIGN-3D, 2021 que se puede observar en la figura 3.15.

Este modelo fue importado en Unreal Engine, y de las animaciones que trae, nos quedamos con el bucle de la caminata, dicho modelo y animación fueron agregados a un personaje “AntCharacter” dentro del proyecto del simulador.

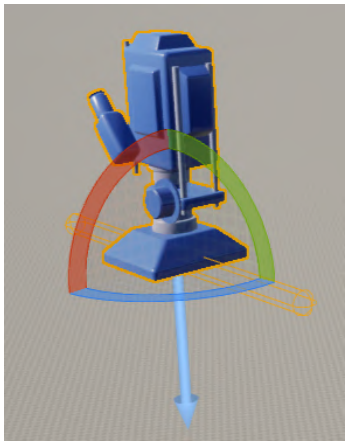
Figura 3.15: Modelo 3D de la hormiga

3.2.3. Movimiento de la cámara

Diseñamos los movimientos del simulador para que se asemejen a los del robot físico, el cual se desplaza hacia adelante y realiza giros con un ángulo hacia la izquierda o la derecha. En cada paso, utilizamos un vector de avance unitario *(Forward vector)*, un vector horizontal unitario *(Right vector)*, y una rotación en el eje horizontal *(Yaw Input)*, como se pueden ver en las figuras 3.16a y 3.16b. Para cada movimiento se genera un vector de desplazamiento realizando una combinación lineal entre el vector de avance y el vector horizontal, además se genera una rotación sobre el eje horizontal, de tal forma que cada movimiento se asemeje a los pasos del robot real, en cuanto a distancia recorrida y capacidad de rotación.



(a) El vector rojo representa el Forward Vector, el vector verde representa el Right Vector.



(b) El ángulo azul representa el Yaw Input.

Figura 3.16: Representación de vectores

3.2.4. Utilización de Mass AI para movimiento de hormigas

Para simular el camino de hormigas, utilizamos la herramienta *Mass AI* de Unreal Engine 5 que permite crear y simular grandes multitudes de personajes no jugables de manera eficiente y realista. Esta es la única alternativa actualmente que permite simular multitudes con rutas predefinidas, que eviten obstáculos, y permitan reaccionar dinámicamente.

Utilizamos una herramienta para trazar el camino que deben seguir las hormigas y otra para agregar las instancias de las hormigas deseadas. También agregamos la configuración de como debe actuar el conjunto de instancias de hormigas, estableciendo características como la velocidad, su aceleración máxima, y reglas de evitar colisiones entre hormigas, como lo son la distancia de detección de obstáculos, la distancia de separación de obstáculos y la distancia de evasión predictiva. Los valores iniciales por defecto de la herramienta simulan la caminata de humanos, por lo que dichos valores fueron ajustados al tamaño y contexto de las hormigas.

En la figura 3.17 se muestra como se trazan caminos de ambos sentidos los cuales las hormigas recorrerán.

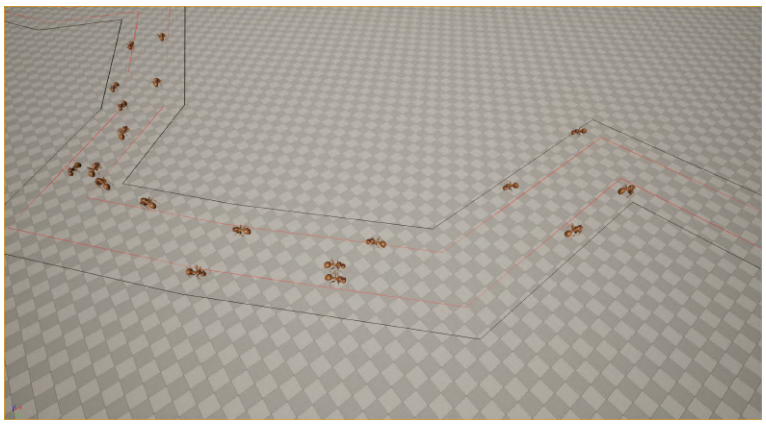


Figura 3.17: Hormigas agregadas por MassSpawner en el suelo, recorriendo el camino.

3.2.5. Agregando detalle al suelo

Para las texturas del suelo, utilizamos los escaneos del mundo real `Leafy Grass Texture` y `Dirt Texture` de Charlotte Baglioni Baglioni, 2023, 2024 como texturas planas en el suelo, de dominio público. Sobre estas, agregamos también para las texturas del follaje MegaScan “Grass Clumps” de Quixel Bridge, una colección de contenido dentro de Unreal Engine Quixel, 2023. Su uso es libre siempre que sea dentro

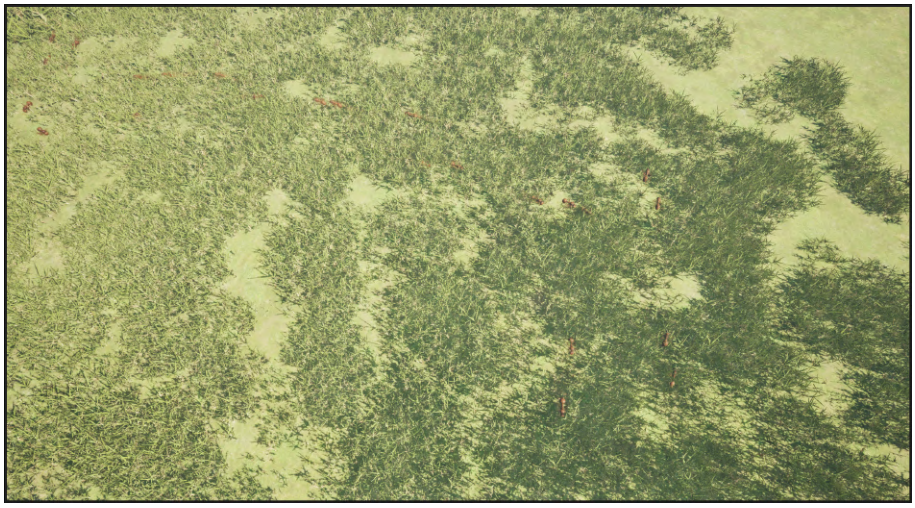


Figura 3.18: Hormigas recorriendo el suelo con pasto

de proyectos de Unreal Engine. Es posible configurar la altura del pasto, el espacio entre cada textura del pasto, las zonas con pasto, entre otras. Puede verse el resultado final en la figura 3.18.

3.2.6. Interfaz de programación

Utilizando un Plug-In denominado "Remote Control API" de Unreal Engine, se habilita el control remoto del proyecto de manera remota a través de solicitudes HTTP por medio de Custom Events. Esto es útil para poder interactuar con la simulación desde Python u otras herramientas.

Agregamos los siguientes *Custom Events* "Socket Move Left", "Socket Move Right" y "Socket Move Forward", estos realizan la misma acción que presionar las teclas A, W o D, permitiendo rotar y avanzar hacia delante, con tiempos tomados de forma que el efecto de tener el vector de movimiento modificado a un valor distinto de cero haga un movimiento pronunciado similar a como lo hace el robot.

Además, incorporamos también el *Custom Event* "Get Frame" que es llamado en la figura 3.19, el cual al ser ejecutado genera una captura con el timestamp actual, además de generar un archivo con el mismo nombre que contiene los Bounding Boxes de la captura en el formato que utiliza YOLO (id, xcenter, ycenter, width, height). Esto permite realizar capturas del simulador ya etiquetadas con las cuales poder entrenar al modelo de detección. Para generar los bounding boxes se proyecta la esfera que encierra a cada hormiga visible por la cámara sobre la imagen renderizada. Luego se toma el centro y radio de esta proyección para construir un rectángulo que encierre a la misma, el cual es nuestro bounding box resultante para la hormiga. Puede verse un ejemplo de su resultado en la figura 3.20.

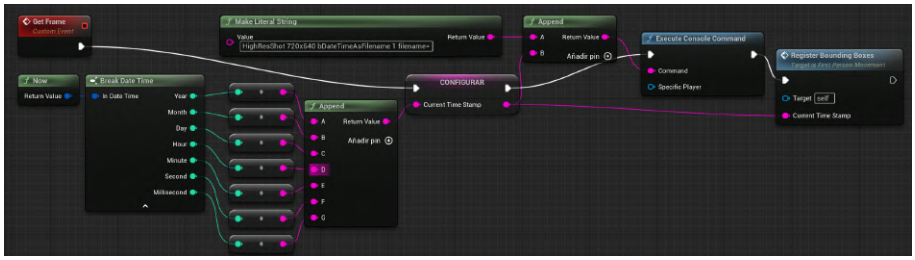


Figura 3.19: Grafo de evento de Get Frame en FirstPersonMovement para ser accedido por API REST

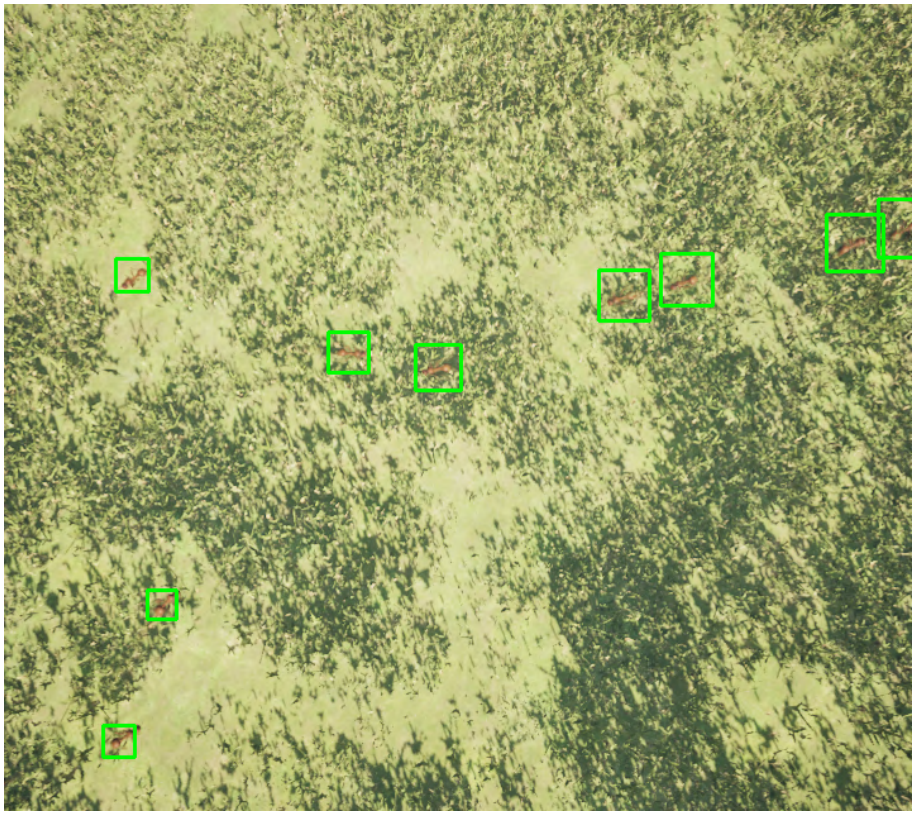


Figura 3.20: Captura del simulador con sus bounding boxes hecha con Get Frame

3.3. Modelo

La detección de hormigas es una tarea que presenta grandes limitantes en el tipo de sensores que pueden ser utilizados para este propósito. Dado el pequeño tamaño de las hormigas, el único sensor capaz de detectarlas que disponemos son cámaras.

El utilizar una cámara nos permite obtener información sobre las hormigas individuales, ya que pueden distinguirse en la imagen, dándonos así más herramientas para abordar el problema.

Se puede considerar el problema como uno de clasificación sobre las imágenes para obtener la dirección de movimiento del robot, que tiene como dificultad la poca disponibilidad de datos, al no ser común los datasets que etiqueten imágenes de hormigas con la dirección del camino.

Por otra parte, se podría identificar como un problema de clasificación de objetos, para el cual si hay disponible una considerable cantidad de datatsets etiquetados, y a partir de las posiciones de las hormigas detectadas utilizar otro algoritmo, ya sea una heurística o modelo de inteligencia artificial, para decidir la dirección que debe tomar el robot para seguir o encontrar el camino.

Otra opción para analizar las imágenes consiste en utilizar bibliotecas que implementen detección de objetos a partir de los colores, pero esta opción fue descartada debido a que el contraste entre las hormigas y el terreno normalmente es muy bajo como para producir resultados satisfactorios.

Teniendo estas consideraciones en mente se decide utilizar una cámara y desarrollar un modelo de visión que detecte las hormigas en la imagen, e inicialmente utilizar una heurística para decidir los movimientos del robot en base a las detecciones.

3.3.1. Hardware

En esta sección se utiliza una PC de escritorio para el entrenamiento y evaluación del modelo. Esta posee un procesador Intel i5-3570 @ 3.40GHz, una tarjeta de video NVIDIA GeForce GTX 1060 6GB y memoria RAM DDR3 12GB.

3.3.2. Dataset

En esta sección presentamos los distintos conjuntos de imágenes etiquetadas que disponemos para el entrenamiento del modelo de inteligencia artificial. Estos consisten de un conjunto público obtenido de Roboflow, un conjunto propio obtenido en campo, y un conjunto sintético desarrollado en nuestro simulador.

Dataset con imágenes de la revisión de antecedentes

El dataset creado se compone de imágenes etiquetadas de hormigas obtenidas a partir de otros datasets de público acceso disponibles en Roboflow. Estas imágenes consistían de fotos tomadas a una distancia relativamente cercana a las hormigas, algunas a distancia similar a las distancias manejadas por nuestro robot, y en su mayoría en entornos naturales como en tierra, pasto, hojas, etc., aunque también se encontró en los datasets una presencia no menor de imágenes en entornos de laboratorio, no naturales a las hormigas y por ende no similares a aquellos que nuestro robot se encontraría, como son superficies claras de mesas de laboratorio, recipientes, entre otros instrumentales, además de imágenes de hormigas tomadas a una distancia muy cercana. Dado que muchas de estas imágenes no se asemejan lo suficiente al contexto real en cuanto a posición de cámara, distancia y entorno, generamos alternativas para poder entrenar el modelo en condiciones que se asemejen más al uso práctico.

El tamaño de este dataset es de 21.846 imágenes, separándose 17.534 para el conjunto de entrenamiento y 4.312 para la validación.

Dataset sintético

Utilizando el simulador desarrollado (3.2) y un script en Python que obtenía capturas etiquetadas a través de una API REST, se creó un dataset con fotogramas del simulador, situando la cámara en una posición similar a la del robot en los escenarios reales. Este dataset contiene varios tipos de fondos, como pasto de diferentes alturas, caminos entre tierra y superficies sin vegetación, con el objetivo de aumentar la cantidad de imágenes representativas del contexto de uso real, este dataset cuenta con un total de 5.107 imágenes etiquetadas.

Dataset hecho en CVAT

Fue creado utilizando CVAT (Computer Vision Annotation Tool), una herramienta de etiquetado manual. Este conjunto se compone de fotogramas de videos capturados por nosotros, tomados desde la cámara instalada en el robot, para obtener imágenes reales que representen la posición, altura y ángulo que tendrá la cámara en el entorno real. Esto proporciona un contexto más preciso para la detección de hormigas en condiciones similares a las del uso práctico del modelo.

Al ser las hormigas muy pequeñas se tomaron fotogramas con una resolución de 1280x852, permitiendo que las hormigas etiquetadas ocuparan cuadros de entre 30 y 70 pixeles de ancho o largo (dependiendo de la orientación), lo cual facilita el poder detectarlas con los modelos de visión descritos en la sección 3.3.3.

Este conjunto de datos tiene la ventaja de estar ajustado al entorno específico donde se realizarán las pruebas, ya que las imágenes fueron tomadas directamente de ese contexto. Sin embargo, presenta la desventaja de no incluir hormigas de distintos tamaños ni de otros entornos diferentes a los considerados en nuestro proyecto. Esto limita su utilidad al momento de generalizar los resultados a situaciones fuera del proyecto, como el uso de cámaras diferentes, variaciones en la posición de las mismas, o diversidad en los tamaños, tipos de hormigas y características de los suelos.

El tamaño de este dataset es de 2.975 imágenes, separándose 2.679 para el conjunto de entrenamiento y 296 para la validación.

3.3.3. Modelos de Visión

Para la detección de hormigas en las imágenes se estudiaron diferentes arquitecturas, considerando su aplicabilidad para la tarea (en este caso detección de objetos), rendimiento y recursos necesarios tanto para el entrenamiento como para su funcionamiento, ya que debía ser capaz de ejecutarse en el hardware disponible en conjunto con el resto del sistema. Se evaluaron distintos modelos ampliamente utilizados (lo que nos otorgaba una mayor disponibilidad de información sobre los mismos para una comparación, y documentación y respuestas a consultas generales en caso de seleccionarlo) como MobileNet, ResNet y Yolo.

En nuestra comparación de los modelos destacamaos que MobileNet presentaba un menor requerimiento de recursos, aunque alcanzando un menor desempeño. Por otro lado, si bien los requerimientos de Yolo y ResNet son mayores, su utilización es factible con los recursos disponibles (placa odroid N2+, con CPUs ARM de 2.2Ghz, y RAM DDR4 de 1320Mhz) consiguiendo un mayor rendimiento. De estas últimas dos Yolo destaca al ofrecer varias versiones de su modelo, que nos ofrecen un mejor balance entre tamaño/requisitos y el rendimiento esperado, además de disponer de una API que facilita el uso del modelo tanto en el entrenamiento como en la inferencia, implementando estas partes, evitando posibles errores y optimizando su uso.

De todas las opciones evaluadas Yolo fue la que mejor se ajustaba a todas estas necesidades, específicamente eligiéndose YOLOv8 mediante la API de Ultralytics, la arquitectura más avanzada de YOLO al momento de iniciar esta etapa del proyecto.

Yolo

Con el fin de hacer un mejor aprovechamiento de los recursos y datos disponibles, se eligió utilizar un modelo pre entrenado del modelo yolov8n, el más pequeño de YOLOv8, que nos proporciona una mayor cantidad de cuadros por segundo con un menor consumo de hardware, y manteniendo una complejidad suficiente para nuestra tarea.

Utilizando nuestro propio dataset (3.3.2) se entrenó por 50 épocas, deteniéndose utilizando *early stopping*, obteniendo un mAP50 de 98.62 %, mAP50-95 de 68.83 %, precisión 95.57 % y recall 96.57 %, sobre el conjunto de validación, y detectando en promedio el 40 % de las hormigas en otros videos obtenidos luego de construir el dataset y realizado el entrenamiento. Esto nos indica un claro sobreajuste, producido por el hecho de que las imágenes, tanto de entrenamiento como de validación, provienen de los mismos dos videos, y no todos los frames presentan una diferencia significativa con los demás.

Este sobreajuste no presenta un problema, ya que el proyecto no se enfoca en obtener una detección perfecta de todas las hormigas, sino en detectar una cantidad suficiente como para determinar la dirección del camino.

Evaluando nuestro nuevo modelo en un video del mismo entorno, pero que no participó del dataset de entrenamiento (no se encuentra en los conjuntos de entrenamiento ni validación, por lo que no ha sido visto por el modelo), corroboramos que se obtienen resultados satisfactorios para nuestra tarea (cercano al 40 %), detectando de forma precisa suficientes hormigas en cada imagen, con entre 2 y 5 detecciones, y sin generar falsos positivos, lo que nos da la capacidad de definir el camino en la imagen, dado el pequeño segmento observado del mismo y el hecho de que los caminos no suelen tener curvas demasiado bruscas, este se acerca en su forma a una recta, por lo que una mayor cantidad de hormigas detectadas en la imagen sería redundante para definir el camino.

Cabe aclarar que el modelo fue entrenado con un dataset representativo del entorno al que teníamos acceso, compuesto por caminos marcados de hormigas sobre la tierra entre un pasto corto, con una iluminación leve en el horario de tarde/noche y algo de iluminación artificial.

Si bien estas condiciones se asemejan a las de otras huertas y cultivos que pudimos visitar, para lograr los mismos resultados en otros entornos, ya sea con distintas hormigas, mayor iluminación o suelos que no sean tierra oscura, sería necesario contar con imágenes etiquetadas de los mismos para ampliar el dataset y reentrenar el modelo.

Utilizando el dataset con imágenes del estado del arte (3.3.2) se entrenó un modelo, también partiendo de yolov8n pre entrenado, por 100 épocas. Al ser este dataset mucho mayor en comparación y con imágenes diversas en luminosidad, tamaños, tipos de hormigas y suelos, considerábamos podría generar un modelo más general para distintos ambientes.

Al finalizar el entrenamiento las métricas obtenidas fueron mAP50 de 83.19 %, mAP50-95 de 44.44 %, precisión 84.93 % y recall 76.19 %. El hecho de que los números sean menores que en el entrenamiento anterior no indican que el modelo sea peor, ya que anteriormente se presentaba un sobreajuste al entorno.

Evaluando el modelo en videos tanto en campo como en laboratorio, notamos que generaba una muy buena detección de las hormigas en laboratorio (ambientes de hormigas impresas o sobre fondo blanco) detectando todas las hormigas presentes, y detectando la gran mayoría de hormigas en ambientes reales sobre baldosas u hormigón. Ambos entornos presentan un alto contraste entre la hormiga y el suelo.

Al evaluar el modelo sobre tierra, se observó que no logra detectar hormigas en ninguna condición de suelo o luminosidad. Si bien esto representa una limitación importante para nuestro entorno objetivo, el modelo demostró ser útil como complemento del modelo original, permitiendo ampliar la capacidad de detección a superficies de alto contraste como baldosas y hormigón, donde nuestro modelo inicial no había sido entrenado.

3.4. Integración del robot con el modelo

Después de desarrollar un modelo que detectara exitosamente hormigas en diversos entornos y de perfeccionar el movimiento del robot hexápodo para asegurar una caminata estable y eficiente, procedimos a combinar ambas tecnologías. Este proceso consistió en integrar el sistema de detección con el robot, permitiéndole identificar y reaccionar ante la presencia de hormigas en tiempo real. La unión de estas dos partes permitió que el robot no solo se desplazara de manera autónoma, sino que también interactuara con su entorno de manera inteligente.

3.4.1. Obtención de imagen

Para obtener la imagen de la cámara se utilizó la librería de python *opencv* Doxygen, 2024 que permite obtener capturas de la cámara a demanda.

Para la detección de hormigas se utilizó el modelo entrenado que fue explicado en la sección 3.3. Este modelo es cargado y luego, con la función *predict* de la librería YOLO el modelo procesa la imagen de la cámara y nos devuelve los *bounding boxes* correspondientes a cada hormiga detectada.

A partir de los *bounding boxes* se calcula el punto medio de cada uno y estos puntos son utilizados como entrada para el algoritmo de mínimos cuadrados, que nos devuelve la recta que mejor aproxima a los puntos. El método de mínimos cuadrados es una herramienta ampliamente utilizada en diversos campos, como la ingeniería, la física y la economía, debido a su capacidad para ajustar una línea o curva a un conjunto de datos de manera óptima, minimizando el error cuadrático. Se eligió este método porque, dado el campo de visión limitado del robot y su proximidad al suelo, se asume que los caminos generados por las hormigas tendrán una distribución aproximadamente lineal. Además, su robustez y simplicidad lo hacen ideal para este tipo de análisis.

3.4.2. Toma de decisión

Luego de obtener la recta que proporciona el algoritmo se analiza qué acción debe tomar el robot, para ello se construyó una heurística que dada la información de la trayectoria de las hormigas decide el próximo paso del robot. Se plantearon y probaron dos heurísticas que serán explicadas a continuación.

Primer heurística

Para la primera heurística, se decidió dividir la imagen en tres secciones: izquierda, centro y derecha. Esta segmentación permite identificar de manera más clara en qué área de la imagen se encuentra el punto medio de la recta de mínimos cuadrados. Luego de definir estas tres áreas, se establecieron puntos de corte específicos para determinar los límites de cada sección. De esta manera, se puede comparar la posición del punto medio en el eje x con estos puntos de referencia para identificar si está en la parte izquierda, central o derecha de la imagen. Se marcaron los puntos de corte en las coordenadas correspondientes a un tercio del ancho de la imagen y dos tercios del ancho de la imagen, separando la imagen en tres partes de anchos iguales.

Si la coordenada x del punto medio caía en la parte izquierda, el robot giraba a la izquierda, si caía en el medio se avanzaba y si caía a la derecha giraba a la derecha. Se puso esto a prueba y se detectó que en algunos casos no tomaba buenas decisiones; si tenemos un camino de hormigas con una inclinación clara hacia la derecha pero que están en la esquina izquierda de la imagen como se puede ver en la figura 3.21, el robot optará por girar a la izquierda, lo que puede causar que pierda de vista a las hormigas, ya que estas se están desplazando hacia la derecha. Es por esto que se decidió complejizar la heurística para contemplar estos casos.

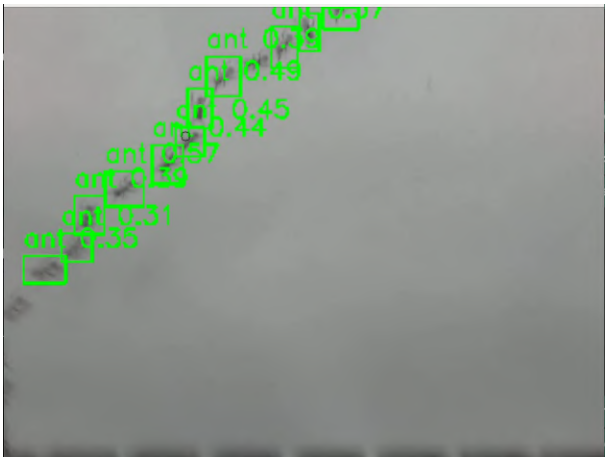


Figura 3.21: Hormigas detectadas por el modelo

Segunda heurística

En esta heurística, además de considerar el punto medio de la recta obtenida mediante mínimos cuadrados, también se toma en cuenta su inclinación. Dado que las pendientes de las líneas verticales tienden a infinito, se llevaron a cabo pruebas experimentales para determinar un valor suficientemente grande que permita clasificar una recta como "prácticamente vertical". Los resultados indicaron que las pendientes de estas líneas, dependiendo de una leve inclinación hacia la izquierda o derecha, alcanzaban valores absolutos cercanos a 4,5 (positivos o negativos según la inclinación).

Por otro lado, dado que las pendientes de las líneas horizontales son iguales a cero, se realizaron pruebas experimentales para identificar un umbral lo suficientemente pequeño como para clasificar una recta como "prácticamente horizontal". Estas pruebas concluyeron que una pendiente con un valor absoluto cercano a 0,5 puede considerarse como "prácticamente horizontal".

Con base en estos valores, se desarrolló el siguiente razonamiento:

- **Caso de recta vertical**: Si la recta es prácticamente vertical, se aplican los razonamientos de la primera heurística definida. En esta, la decisión depende de la coordenada x del punto medio. Si el punto medio está del lado izquierdo de la imagen, se realiza un giro antihorario; si está en el lado derecho, un giro horario; y si se encuentra cerca del centro, se procede a avanzar.
- **Caso de recta inclinada hacia la derecha**: Cuando la pendiente indica que el camino de las hormigas está inclinado hacia la derecha, la imagen se divide de manera no uniforme: en lugar de dividirla en tres partes iguales, el extremo izquierdo se subdivide en dos regiones. Así, se definen tres zonas:
  - **Significativamente hacia la izquierda**: Si el punto medio cae en esta región, se realiza un giro antihorario.
  - **Moderadamente hacia la izquierda**: Si el punto cae en esta zona, se procede a avanzar.
  - **Central o a la derecha**: Si el punto está en alguna de estas regiones, se realiza un giro horario.

  Esta estrategia asegura que si las hormigas se desplazan hacia la derecha pero su punto medio indica una concentración hacia la izquierda de la imagen, no se pierda su trayectoria al realizar un giro horario.
- **Caso de recta inclinada hacia la izquierda**: De forma análoga al caso anterior, cuando la pendiente indica que el camino de las hormigas está inclinado hacia la izquierda, se aplica una subdivisión no uniforme en el extremo derecho. Así, se definen las siguientes zonas:
  - **Significativamente hacia la derecha**: Si el punto medio cae en esta región, se realiza un giro horario.
  - **Moderadamente hacia la derecha**: Si el punto cae en esta zona, se procede a avanzar.
  - **Central o a la izquierda**: Si el punto está en alguna de estas regiones, se realiza un giro antihorario.

Esto garantiza que, si las hormigas se desplazan hacia la izquierda pero su punto medio está concentrado en el lado derecho de la imagen, no se pierda su trayectoria por realizar un giro anithorario.

- **Caso de recta casi horizontal**: Si la pendiente es cercana a cero, la recta se considera prácticamente horizontal. En este caso, se eligió arbitrariamente girar en sentido horario para comenzar a seguir a las hormigas en esa dirección.

Esta metodología busca maximizar la capacidad de seguir el camino de las hormigas mientras se adapta a las distintas inclinaciones de la trayectoria. En la figura 3.22 se puede ver un diagrama de flujo para le heurística planteada.

Figura 3.22: Diagrama de flujo representando la toma de decisiones del movimiento

A continuación se muestra el pseudo-código de la heurística, siendo x la coordenada horizontal del punto medio de la recta de mínimos cuadrados y *pendiente* la pendiente de la recta de mínimos de cuadrados.

```
# Evaluar la pendiente
si pendiente es vertical:
    si x esta centrada:
        dar_paso_adelante()
    sino:
        si x esta a la izquierda:
            giro_antihorario()
        sino:
            giro_horario()

sino si pendiente es hacia la derecha:
    si x esta significativamente a la izquierda:
        giro_antihorario()
    sino si x esta moderadamente a la izquierda:
        dar_paso_adelante()
    sino:
        giro_horario()

sino si pendiente es hacia la izquierda:
    si x esta significativamente a la derecha:
        giro_horario()
    sino si x esta moderadamente a la derecha:
        dar_paso_adelante()
    sino:
        giro_antihorario()

sino: # pendiente horizontal
    giro_horario()
```

Listing 3.1: Código de heurística

Las pruebas realizadas con la nueva heurística fueron exitosas dado que el robot fue capaz de seguir distintas configuraciones de caminos de hormigas en un entorno de laboratorio, tanto en los casos en los que la primera heurística tuvo éxito como en los que no.

Capítulo 4

Experimentación

En este capítulo se evalúa el desempeño del robot desarrollado, junto con la heurística y el modelo de detección de hormigas implementado. El objetivo principal es determinar si el robot es capaz de operar eficazmente en un entorno real, así como analizar la eficacia tanto de la heurística como del modelo de detección.

Para estas pruebas elegimos el modelo que nos devolvió las mejores métricas y que tuvo el mayor éxito a la hora de reconocer hormigas, planteado en la sección 3.3.3, la heurística que fue mostrada en la sección 3.4.2, y el modelo de robot que presentó mayor estabilidad y una menor desviación según los experimentos presentados en la sección 3.1.2

4.1. Pruebas en ambiente de laboratorio

Para probar exclusivamente la heurística, diseñamos un entorno controlado específicamente para este propósito, configurado para garantizar condiciones óptimas y aisladas de factores externos que pudieran interferir con los resultados.

Es por esto que se realizó esta prueba en el laboratorio, utilizando un piso cubierto de alfombra. Esta elección se basó en que en varias de las pruebas previas de las caminatas del robot se utilizó esta superficie y se observó que no presentaba inconvenientes para el desempeño de sus movimientos.

Para esta prueba se utilizaron impresiones de caminos de hormigas sobre fondo blanco, con una gran densidad de hormigas que probamos que nuestro modelo puede detectar, una variante puede verse en la figura 4.1. Los caminos fueron dispuestos de manera de generar distintas configuraciones para cada corrida del robot, que evalúen las transiciones entre estados descrita en 3.22

Figura 4.1: Camino impreso, uno de nueve variantes que se imprimieron

Se iniciaron todas las pruebas de forma que el robot pudiera visualizar las hormigas desde el primer momento de la prueba.

Se realizaron cinco corridas y en todas el robot pudo seguir exitosamente el camino, sin perderlo de vista en ningún momento. Las configuraciones de los caminos consistieron de:

- Un camino recto.
- Un camino curvo hacia la derecha (en forma de elipse).
- Un camino curvo hacia la izquierda (en forma de elipse).
- Un camino con una curva a la izquierda y una a la derecha.
- Un camino que combina varias curvas en ambos sentidos y segmentos rectos.

En la figura 4.2 se pueden observar imágenes de la detección que realiza el modelo sobre la representación de las hormigas. Cada recta dibujada tiene además en el centro la decisión que tomó el modelo, con las opciones de dar un paso hacia la izquierda *('Left (L)')*, derecha, *('Right (R)')*, y adelante *('Forward (F)')*.

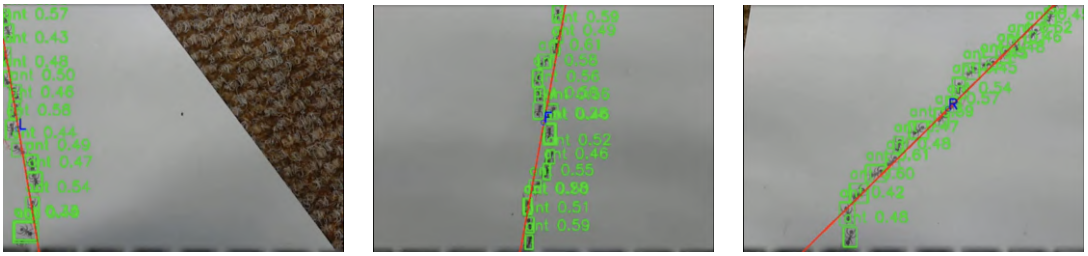


Figura 4.2: Detección de hormigas

Teniendo en cuenta los resultados obtenidos, dónde más del 90 % de las corridas fueron exitosas, se puede afirmar que la heurística demostró ser efectiva y confiable. Cuando el contraste entre el fondo y las hormigas es elevado, el modelo logra identificarlas con precisión, y la heurística permite seguir el camino sin perderlo de vista en ningún momento. Estos resultados destacan que, en escenarios con una cantidad significativa de hormigas detectadas, la heurística no solo funciona correctamente, sino que se comporta de manera robusta y eficiente, confirmando su utilidad para el propósito planteado.

4.2. Prueba en camino sobre baldosas

Con el objetivo de evaluar cómo nuestro modelo detecta hormigas reales en un entorno fuera del laboratorio, y analizar el desempeño de la heurística en función de la cantidad de hormigas que el modelo es capaz de identificar, llevamos a cabo las siguientes pruebas.

Dado el resultado de la prueba anterior 4.1, el siguiente paso en complejidad para nuestra detección de hormigas es pasar de caminos de hormigas impresas estáticas con fondo blanco de gran contraste, a caminos de hormigas reales en movimiento con fondo gris de gran contraste, como lo son las baldosas que utilizamos.

Para este ambiente se realizaron distintas pruebas donde el robot, en distintas posiciones, procesaba las hormigas que estaba viendo y teniendo en cuenta la heurística definida tomaba la decisión de qué movimiento realizar. A continuación se pueden ver una de estas pruebas en la figura 4.3.

Figura 4.3: Detección de hormigas sobre baldosas claras. Las detecciones están marcadas por el rectángulo verde con su porcentaje de confianza, la linea roja es la recta que mejor se ajusta a las detecciones y la R azul indica el movimiento escogido, en este caso derecha (*Right*)

En la imagen se pueden ver las hormigas detectadas. Al momento de realizar la prueba tuvimos disponibles una cantidad de hormigas mucho menor que las que teníamos impresas en la prueba anterior, al no ser posible encontrar una cantidad similar, por lo que en la mayoría de los casos la heurística tuvo que decidir su próximo paso únicamente con la información de posición de una única hormiga. Aún así se puede observar que el modelo tomó decisiones coherentes con respecto a la posición de la hormiga que identificó.

Con los resultados obtenidos en esta prueba podemos concluir que el modelo puede detectar hormigas reales en movimiento, y que la heurística con menos información de hormigas detectadas puede tomar decisiones correctas.

4.3. Prueba sobre tierra y pasto

La siguiente prueba se enfocó en aumentar la complejidad del entorno en el que se encuentran las hormigas, avanzando hacia escenarios más representativos del contexto agrícola que buscamos abordar. En esta etapa, evaluamos el desempeño del modelo utilizando hormigas caminando sobre pasto o tierra, ambientes más cercanos a su entorno natural en comparación con las baldosas grises. El objetivo fue analizar cómo nuestro modelo detecta hormigas en condiciones de menor contraste entre ellas y el fondo, y una menor iluminación, con un camino de hormigas con una densidad mayor a las pocas hormigas que encontramos en las baldosas.

Para estas pruebas utilizamos videos previamente grabados de caminos de hormigas sobre pasto y tierra, ya que no fue posible realizar pruebas en vivo debido a la ausencia de caminos de hormigas en el momento de la evaluación. Los videos fueron capturados desde el robot, en distintos ángulos, en diversos escenarios y con diferentes condiciones de iluminación, con el objetivo de abarcar una mayor variedad de situaciones.

En los videos analizados, cada fotograma contenía entre 10 y 12 hormigas, de las cuales el modelo logró detectar consistentemente entre 2 y 5 hormigas.

A continuación se pueden ver algunas de estas pruebas en la figura 4.4

Figura 4.4: Detección de hormigas sobre tierra y pasto. Las detecciones están marcadas por rectángulos verdes con su porcentaje de confianza, las líneas rojas son las rectas que mejor se ajustan a las detecciones y las letras azules indican el movimiento escogido, izquierda (L), adelante (F) y derecha (R), *Left*, *Forward* y *Right* respectivamente

Se analizó el comportamiento de la heurística basándose en la información proporcionada por el modelo. Las imágenes muestran claramente las hormigas detectadas y las decisiones tomadas por la heurística en cada situación.

A partir de los resultados, concluimos que nuestro modelo es capaz de detectar aproximadamente el 40 % de las hormigas presentes en un camino de hormigas real estándar. Este nivel de detección menor se debe a las dificultades de contraste e iluminación mencionadas, pero resulta suficiente para que la heurística tome decisiones acertadas y cumpla con su propósito de seguimiento.

4.4. Prueba Final - Simulador Unreal Engine

4.4.1. Introducción

Luego de evaluar la heurística y el modelo de visión en un ambiente controlado como el laboratorio y sobre videos de caminos de hormigas reales decidimos proceder con una prueba que evalúe todas las partes del sistema: movimiento del robot, modelo de detección y heurística, trabajando de forma coordinada en un entrono que presente los desafíos a los que se enfrentaría el robot al utilizarse en el agro. Dadas las dificultades en disponer de un espacio real donde llevar a cabo estas pruebas, se adaptó el simulador desarrollado en 3.2 para incorporar los errores físicos del robot y de detección del modelo de visión, según las mediciones obtenidas en las pruebas anteriores de cada una de estas componentes. Esto no solo permite evaluar el sistema con un nivel de realismo comparable al de realizar las pruebas en un camino de hormigas, sino que también ofrece la posibilidad de amplificar intencionalmente los errores para analizar la robustez del sistema en condiciones extremas y escenarios que podrían no haberse manifestado en las pruebas previas.

4.4.2. Ajustes de Unreal

Para lograr que el simulador se asemeje más a la realidad, introducimos error en la caminata, para que los pasos no sean todos iguales. Esto fue realizado con el objetivo de compensar el hecho de no tener desniveles en el suelo de Unreal y dado que los motores del robot en la realidad pueden tener mínimas variaciones de comportamiento.

Cada paso, utilizaba un vector de avance unitario *(Forward vector)*, un vector horizontal unitario *(Right vector)*, y una rotación en el eje horizontal *(Yaw Input)* como se menciona en la sección 3.2.3, generando una combinación lineal entre los vectores, y una rotación sobre el eje horizontal. En cada uno de estos componentes, se introduce un error, elegido aleatoriamente dentro de un rango, el cual simula el error en los movimientos producido en la interacción entre el robot y el ambiente.

Dado que la variación entre los pasos del robot real no es tan grande, limitamos el rango de variación en el simulador para que estos movimientos no sean excesivamente erráticos. En el paso hacia adelante se introduce un error de $\pm 3\,\%$ en el módulo de la componente del vector de avance, lo que genera pasos ligeramente más cortos o más largos. En el vector horizontal, se introduce un error de $\pm 10\,\%$ respecto al módulo de la componente del vector de avance, lo que genera pasos con mayor o menor desviación. Adicionalmente, la dirección de la cámara rota aleatoriamente entre -2^{o} y 2^{o}.

Los pasos hacia la izquierda y hacia la derecha implican un movimiento diagonal, ya que combinan un avance hacia adelante con un desplazamiento horizontal. Durante estos pasos, se introduce un error tanto en el vector horizontal como en el vector de avance de $\pm 10\,\%$. Adicionalmente, la dirección de la cámara rota aleatoriamente entre 0^{o} y 10^{o} en la dirección del giro. Estos valores fueron escogidos observando los resultados de la tabla A.2 para la velocidad y el time.sleep elegidos.

Una demostración que ilustra el comportamiento de estos movimientos se puede ver en el siguiente video.

4.4.3. Pruebas

Simulación de entorno real

Se construyó un mapa con dos caminos de hormigas unidos en un extremo ubicado en el centro del mismo, como se ve en la figura 4.5, lo que nos ofrece un entorno de evaluación más amplio para las posiciones iniciales del robot. El robot es instanciado en una posición al azar dentro de un área alrededor del centro del mapa, la cual está subdividida en 9 áreas, como se ve en la figura 4.6. Dependiendo de la subárea donde es posicionado, se le asigna un ángulo aleatorio dentro de rangos asignados, con el fin de asegurar que siempre tenga el camino por delante. De esta forma nos aseguramos que todas las posiciones iniciales se dirijan hacia el camino, cubriendo todos los ángulos de entrada por los que el robot podría llegar a él, ya que el modelo cuando no detecta ninguna hormiga, se mueve hacia adelante, y así damos oportunidad que se encuentre con el camino. El tener dos posibles caminos nos permite realizar esta instanciación del robot, la cuál es más realista al cubrir todos los ángulos de ingreso, asemejándose a llegar al camino en un tramo intermedio del mismo, en lugar de llegar en un extremo con una dirección muy similar a la del propio camino.

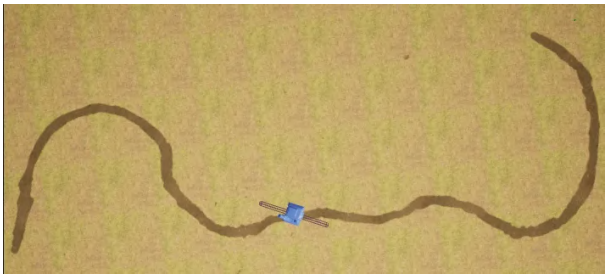


Figura 4.5: Caminos hechos en Unreal. El centro o posición inicial se encuentra en el cuadrado azul (cámara). Las posiciones objetivo del camino se encuentran en los extremos de los mismos.

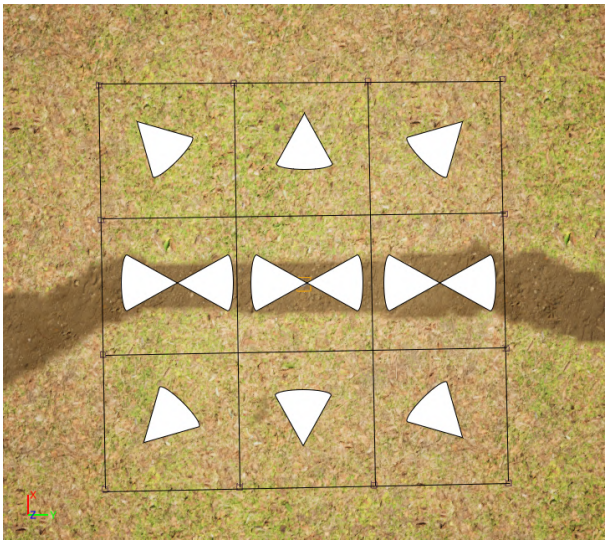


Figura 4.6: Subáreas donde puede instanciarse el robot, con los rangos de los posibles ángulos de orientación

Una vez colocado el robot, se deja que el modelo decida hacia dónde avanzar. Para ello, se capturan fotogramas de la simulación, que son procesados con el modelo presentado en la sección 3.3.3 para detectar las hormigas. Luego, la heurística desarrollada en la sección 3.4.2 decide la dirección en la que debe moverse, basándose en la información procesada. Este proceso se repite hasta completar 50 iteraciones siguiendo cada camino. Las iteraciones se terminan bajo una de las siguientes condiciones:

- **Llegar a una de las metas:** El robot alcanza una de las posiciones objetivo definidas en el mapa, ubicadas en los extremos de los caminos.
- **Perderse del camino:** Si el robot no detecta hormigas durante 10 pasos consecutivos, se considera que se ha desviado del camino. En el simulador esto se implementa finalizando la iteración al realizarse 10 pasos consecutivos hacia adelante, dado que la heurística decide avanzar si no encuentra hormigas.

Dado cualquiera de estos dos casos, se guardan las siguientes métricas de la iteración:

- Si fue exitoso en llegar a una meta
- Posición inicial al comenzar
- Posiciones al realizar cada paso
- Si tomó el camino derecho o el izquierdo

Nuestro modelo, según lo observado en la prueba presentada en la sección 4.3, detecta en escenarios reales el 40 % de las hormigas presentes en la imagen, lo cual suele ubicarse entre 2 y 5 hormigas por

imagen. Sin embargo, al entrenar el modelo con imágenes de hormigas generadas en Unreal Engine, el modelo detecta un número mayor de hormigas en este ambiente, dado que este presenta una mayor calidad de detalles en las hormigas y un mejor contraste e iluminación que los escenarios reales. Para compensar esta diferencia, decidimos aplicar la siguiente estrategia: si se detectan menos de 5 hormigas, el simulador utiliza todas las detecciones. Si se detectan más de 5, el modelo ignora al azar el 60 % de las hormigas. En el entorno de Unreal se pueden observar entre 0 y 12 hormigas en el camino en pantalla, siendo común un número cercano a 6 hormigas. Esto permite que el número de hormigas detectadas en Unreal se mantenga en un rango similar al observado en nuestro modelo en la realidad.

Estudio de casos extremos

Aprovechando la versatilidad del simulador para modelar los errores, decidimos realizar pruebas adicionales donde se simularan entornos más complejos, como ser casos con aún menor contraste entre las hormigas y el suelo, o muy poca iluminación, todas ellas causando que el modelo de detección reconozca menos hormigas. Esto es simulado reduciendo el límite de hormigas máximo reconocido por el modelo, ya que la consecuencia de condiciones aún más adversas es una menor detección de hormigas.

Por esto se redujo la cantidad de hormigas detectadas a números menores que 5, hasta un mínimo de 2, ya que al menos esta cantidad es necesaria para el funcionamiento de la heurística con mínimos cuadrados. Estas pruebas no solo evalúan la robustez del sistema en ambientes difíciles para el modelo de visión, sino que también evalúa la heurística al enfrentarse a caminos con pocas hormigas, ya que además de la baja cantidad, al descartar una cantidad de hormigas, se aumenta la posible dispersión de las mismas en la imagen, lo cuál es una buena representación de lo que ocurre en estos casos. Se puede observar un ejemplo de la prueba realizada con un máximo de 3 hormigas en el siguiente video.

4.4.4. Resultados de las pruebas

Resultados con pruebas de 5 hormigas

Para las pruebas con máximo de 5 hormigas se llegó al objetivo en 46 de 50 corridas en el camino de la izquierda, y en 48 de 50 corridas en el camino de la derecha, lo cual representa un acierto del 94 %, validando el sistema completo (caminata, detección y decisión) bajo las mismas condiciones que un entorno real.

En la figura 4.7 se puede observar una gráfica presentando los resultados de las corridas donde, en cada item, se muestra la cantidad de éxitos que hubo en el camino izquierdo y derecho, y la cantidad de hormigas máxima que permitimos detectar al modelo.

Resultados con pruebas con menos hormigas

Al comparar el rendimiento entre el caso de 5 hormigas y el de 2 hormigas, se observa una disminución moderada en el porcentaje de éxito, pasando del 93 % al 75 %, lo que equivale a una caída del 18 %. Este descenso es relativamente pequeño, especialmente considerando que la cantidad de hormigas detectadas por el modelo se reduce en un 60 % al pasar de 5 a 2. Cabe destacar que el hecho de detectar únicamente 5 hormigas ya representa un escenario extremo descrito en la sección 4.3 con caminos sobre pasto, iluminación tenue y artificial, y bajo contraste entre las hormigas y el suelo. Estos resultados confirman que el sistema mantiene un buen desempeño incluso en condiciones adversas, garantizando su funcionalidad tanto en situaciones habituales como en escenarios desafiantes.

Otros análisis

Cabe destacar que cuando se analizaron las detecciones realizadas por el modelo en la simulación se observaron algunos casos de falsos positivos como se puede ver en la figura 4.8. Esto se acerca a la realidad en donde se pueden encontrar falsos positivos a la hora de detectar hormigas que afectan el rendimiento a la hora de seguir los caminos.

En algunos casos como en las figuras 4.8a y 4.8d se ve que los falsos positivos generan una decisión errónea de movimiento.

También se planteó la posibilidad de evaluar el desempeño del robot en función de su punto de partida, donde se analizaron las posiciones iniciales durante las pruebas, que se pueden observar en la figura 4.9. Este enfoque permite analizar si existen patrones relacionados con la frecuencia de fallos según la subárea donde el robot comienza su recorrido, proporcionando una visión más detallada de su comportamiento en distintas condiciones iniciales.

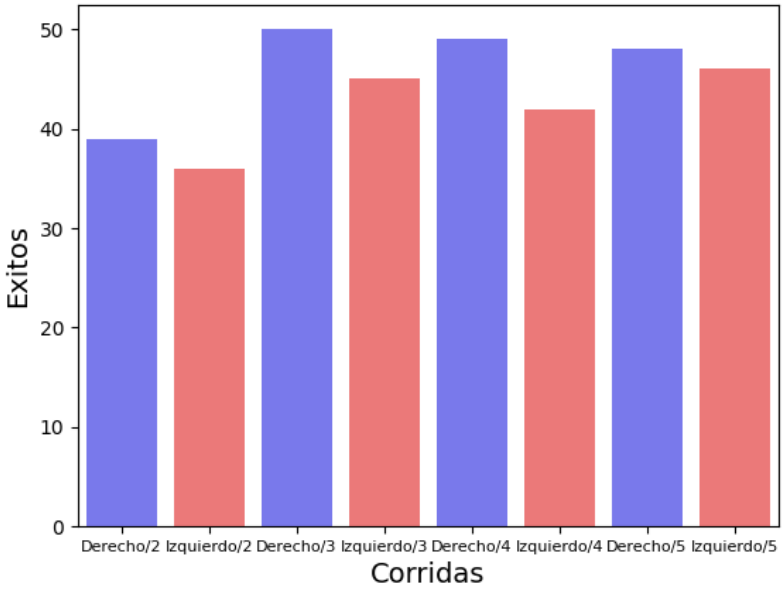
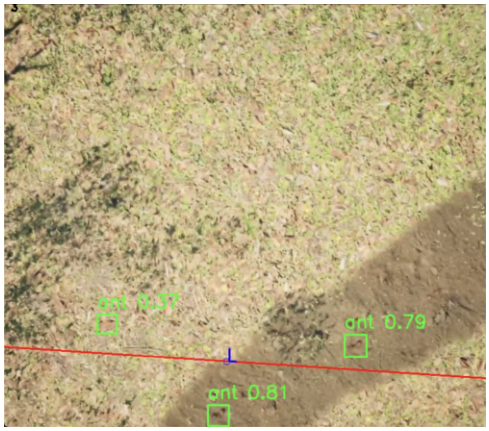
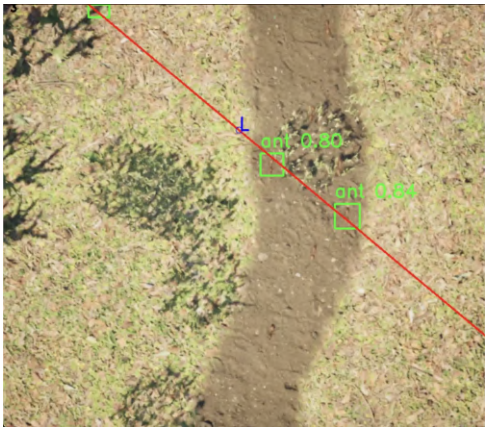


Figura 4.7: Conteo de finalizaciones en el objetivo para el camino de la derecha (Azul) y el camino de la izquierda (Rojo), según la cantidad máxima de hormigas, en 50 corridas para cada camino y cantidad máxima de hormigas

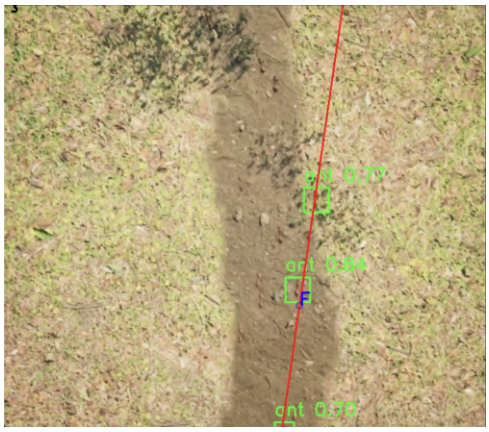
Luego de observar la figura 4.9 se puede concluir que no existe una posición inicial en la que el robot se pierda de manera consistente.



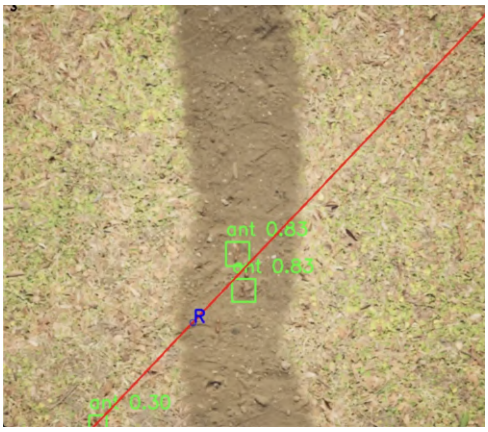
(a) Falso positivo en



(b) Falso positivo



(c) Detección



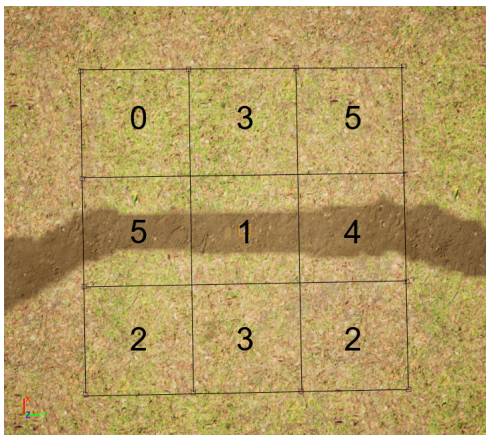
(d) Detección

Figura 4.8: Recopilación de falsos positivos obtenidos en Unreal Engine

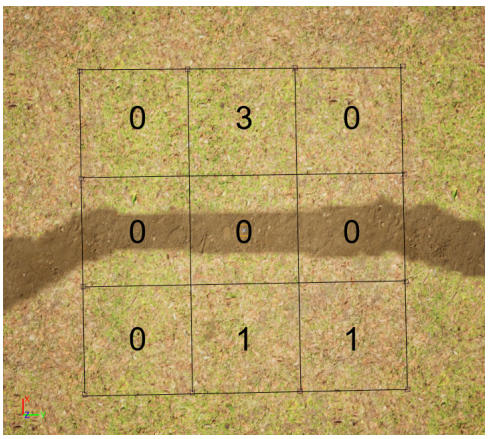
Adicionalmente, al registrar cada posición recorrida en el mapa desde que comienza la iteración hasta que finaliza, podemos generar una representación que muestra todos los caminos recorridos. Esto puede verse representado en la figura 4.10

Al analizar los trayectos representados en la figura 4.10, es notable que, en algunos casos, el robot se desvía cerca de la meta, pasando muy cerca de ella. Aunque técnicamente estos casos se registraron como fallidos, desde un enfoque práctico no pueden considerarse fracasos absolutos, ya que el robot logró recorrer casi todo el trayecto esperado antes de desviarse. Como hipótesis, consideramos que se puede dar porque la meta implementada es un punto que se encuentra un poco antes de terminar el camino de hormigas, con un área alrededor, formando una circunferencia en la cual si el centro de nuestro robot colisiona, se considera exitosa la iteración. En los casos donde se pierde en el final, es posible que por los pasos irregulares quede afuera de dicha circunferencia, y ya sin haber mas camino va hacia adelante hasta llegar a los 10 pasos que equivalen a decir que la iteración fracasó.

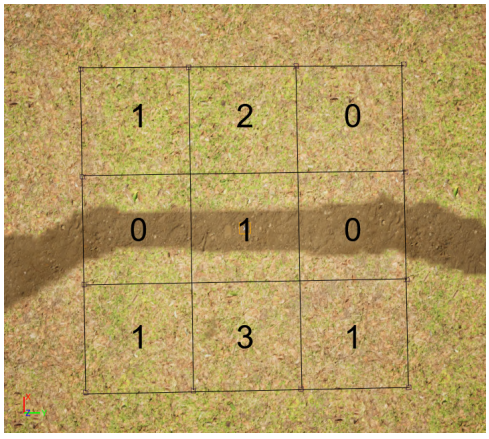
Los casos en los que el robot se desvía del camino suelen ocurrir al inicio del trayecto, o al enfrentar curvas pronunciadas. Además, se observa que la dificultad para seguir el camino aumenta a medida que disminuye la cantidad de hormigas detectadas.



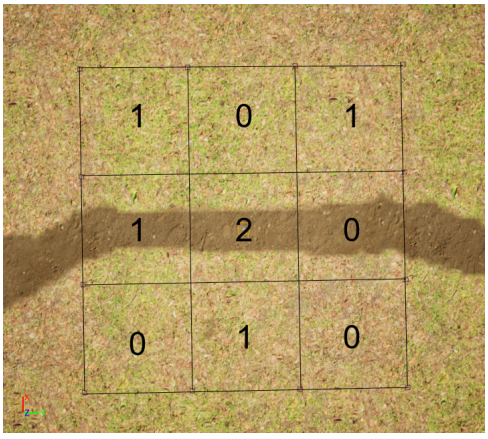
(a) Total: 25 de 100 con máximo 2 hormigas



(b) Total: 5 de 100 con máximo 3 hormigas

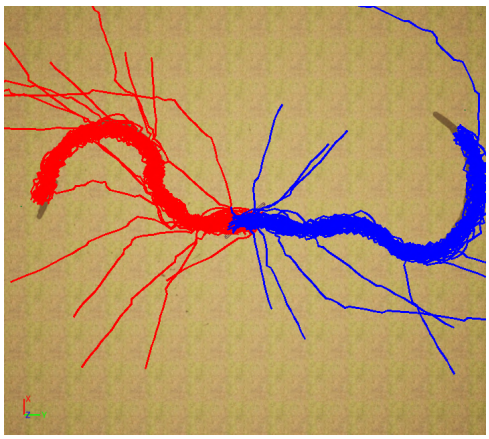


(c) Total: 9 de 100 con máximo 4 hormigas

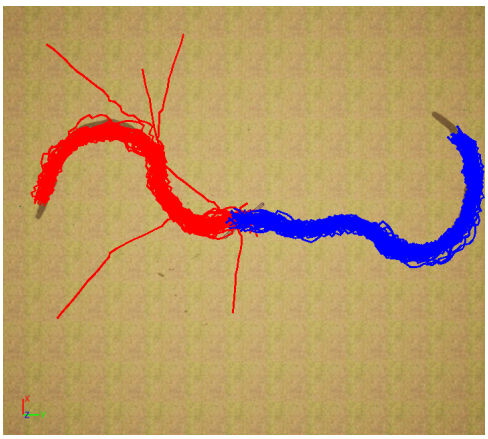


(d) Total: 6 de 100 con máximo 5 hormigas

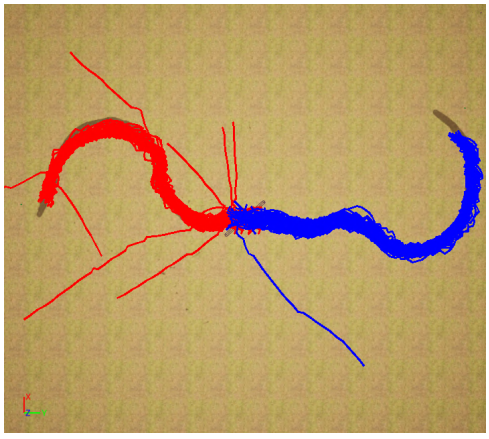
Figura 4.9: Número de corridas donde no se llega a la meta, especificado para cada subárea de inicio



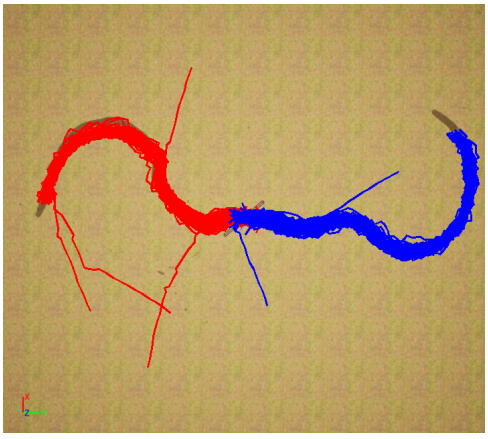
(a) Caminos tomados con máximo 2 hormigas



(b) Caminos tomados con máximo 3 hormigas



(c) Caminos tomados con máximo 4 hormigas



(d) Caminos tomados con máximo 5 hormigas

Figura 4.10: Total de caminos tomados

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

Se desarrolló un robot hexápodo capaz de navegar terrenos irregulares, que detecta caminos de hormigas desde una cámara que tiene sobre sí mismo utilizando un modelo de visión por computadora, con lo que luego decide, por medio de una heurística, el siguiente paso que puede tomar para seguirlas. Además se desarrolló un simulador capaz de representar nuestro robot y los entornos relevantes para el proyecto.

Un aspecto importante de este proyecto ha sido el desarrollo y perfeccionamiento del robot hexápodo, que se diseñó específicamente para operar en terrenos irregulares típicos de los entornos agrícolas. A través de análisis y mejoras en sus patrones de locomoción, logramos un sistema que combina estabilidad y movilidad eficiente. La implementación del patrón de caminata conocido como 'trípode alterno' no solo permitió movimientos estables en superficies complejas, sino que también aseguró que el robot pudiera adaptarse a condiciones desafiantes como pasto largo y terrenos inclinados.

Otro aspecto fundamental de este proyecto fue el entrenamiento de un modelo de visión capaz de detectar hormigas en el agro. Entrenando el modelo concluimos que el contraste entre el color de las hormigas y el fondo juegan un papel fundamental en la efectividad del modelo de detección, al igual que el tamaño de las mismas en la imagen. En entornos con alto contraste, como las baldosas de color gris claro, el modelo mostró capacidad para identificar las hormigas.

Sin embargo, en escenarios más complejos, como aquellos con caminos de hormigas en tierra y pasto, la precisión de la detección disminuyó debido a las variaciones en el fondo y las condiciones de iluminación. No obstante, el modelo aún mostró capacidad para detectar las hormigas y tomar decisiones apropiadas, luego de entrenar el modelo con contextos similares al utilizado para las pruebas.

Con respecto a la heurística, las pruebas en un entorno controlado (baldosas y hojas impresas) revelaron que la heurística desarrollada para tomar decisiones de movimiento del robot fue efectiva. El sistema pudo realizar movimientos coherentes basados en la posición de las hormigas vistas, logrando así seguir los caminos de hormigas encontrados.

Las pruebas realizadas en el simulador permitieron evaluar integralmente el desempeño del sistema como un conjunto en condiciones representativas de escenarios reales. Para ello, se introdujeron ruido y márgenes de error, como errores en la caminata del robot y restricciones en el modelo de detección, replicando las condiciones observadas en pruebas reales. Esto aseguró que los resultados fueran representativos y útiles para identificar fortalezas y debilidades. Este entorno virtual fue clave para integrar y analizar el comportamiento del modelo de detección de hormigas y la heurística, comprobando su efectividad y resiliencia incluso en condiciones adversas. Los resultados destacaron la capacidad del sistema para operar de manera confiable en escenarios desafiantes.

En resumen, el modelo desarrollado ha demostrado ser prometedor en entornos con buen contraste, aunque se enfrenta a desafíos en entornos más complejos. Las pruebas realizadas en el simulador de Unreal Engine confirmaron que el sistema es capaz de mantener un desempeño consistente incluso al introducir errores en el simulador y limitaciones del modelo de detección observadas en condiciones reales. Este entorno virtual no solo permitió evaluar la robustez del sistema bajo escenarios controlados, sino también simular condiciones extremas que reforzaron la adaptabilidad de la heurística y del modelo. Los resultados obtenidos indican que se puede avanzar en el desarrollo de un sistema robusto para la detección de hormigas en diversas condiciones, sentando una base sólida para futuras implementaciones en el campo.

5.2. Trabajo Futuro

A pesar de los avances logrados en este trabajo, existen varias áreas que podrían mejorarse y expandirse en futuras investigaciones:

- Mejorar la detección en entornos con fondos complejos: Se debería investigar el uso de técnicas de preprocesamiento de imágenes, como la eliminación de ruido y el ajuste dinámico de contraste, para mejorar la precisión de la detección de hormigas en superficies no uniformes como tierra o pasto.
- Ampliar las pruebas en entornos naturales: Principalmente se realizaron pruebas con videos grabados. Las pruebas en vivo en condiciones naturales de campo ayudarían a obtener datos más realistas y precisos sobre el comportamiento del modelo en situaciones del mundo real. Resulta muy difícil encontrar un camino de hormigas para hacer pruebas, ya que deben darse de manera natural y por lo cual no fue posible realizar pruebas en espacios no controlados, puesto que a la hora de buscarlas no dimos nunca con un camino de hormigas para usar de prueba.
- Probar nuevas alternativas: Las herramientas como YOLO están progresando de gran manera en muy poco tiempo, y si bien hoy día le cuesta ver hormigas con un fondo que sea principalmente pasto y tierra, es posible que en futuras versiones esto ya no sea una limitante.
- Mejorar el hardware utilizado: Sería beneficioso para la eficiencia del robot reemplazar partes de su estructura por materiales mas robustos, como metales o plásticos mas duros, que lo vuelvan mas resistente al atravesar terrenos irregulares. Además se podrían sustituir la cámara y los motores por modelos de mayor calidad, que aporten mejor información en el caso de la cámara, y en el caso de los motores sean menos susceptibles a fallos. Por último, vemos conveniente agregar una linterna que ilumine a donde esté mirando el robot, para aumentar el contraste, y un LIDAR que permita la implementación de una navegación más segura.
- Explorar la detección de especies de hormigas específicas que afectan la agricultura en Uruguay: Centrarse en identificar y estudiar las especies de hormigas locales que tienen un impacto significativo en la producción agrónoma. Nuestros datasets, entrenamiento de modelo y puesta a prueba fueron basados en imágenes de hormigas en general, sin distinguir entre especies específicas. Trabajamos principalmente con hormigas alrededor del predio de la facultad, sin un enfoque en las especies que afectan de forma directa a los cultivos. Podrían crearse datasets especializados y pruebas específicas orientadas a una o varias especies de hormigas consideradas plagas agrícolas de nuestro país si se cuenta con acceso a estas.
- Generalizar nuestro dataset: Ampliar la diversidad de imágenes etiquetadas incorporando distintos contextos, horarios del día, variaciones de iluminación, estaciones del año, tipos de suelo y variedades de pasto. En general, obtener una mayor diversidad de situaciones para evitar que estas sean una limitante.

En el futuro, la combinación de tecnologías avanzadas como el aprendizaje automático, el procesamiento de imágenes y los sistemas autónomos ofrecerá una gran oportunidad para mejorar la detección y el seguimiento de hormigas en entornos más desafiantes y dinámicos.

Bibliografía

Marković, Ivan, Chaumette, François & Petrović, I. (2014). Moving object detection, tracking and following using an omnidirectional camera on a mobile robot. *IEEE*, 5630-5635.

Imirzian, N., Zhang, Y., Kurze, C., Loreto, R. G., Chen, D. Z., & Hughes, D. P. (2019). Automated tracking and analysis of ant trajectories shows variation in forager exploration. *Nature*. https://www.nature.com/articles/s41598-019-49655-3

Mokariya, L., & Malam, K. (2020). Precision Agriculture - A New Smart Way of Farming. *Agriculture and Environment*, *1*(2), 87-92. https://www.researchgate.net/publication/345807647_Precision_Agriculture_-A_New_Smart_Way_of_Farming

Wu, M., Cao, X., & Guo, S. (2020). Accurate detection and tracking of ants in indoor and outdoor environments. *bioRxiv*.

Zangrandi, M., Arrigoni, S., & Braghin, F. (2020). Control of a Hexapod Robot Considering Terrain Interaction. *None*, 12.

Aakieu. (2021). ax12_control [Visitado el 2024-06-25].

MAXDESIGN-3D. (2021). ANT Model [Visitado el 2024-06-25]. https://sketchfab.com/3d-models/ant-dab7080251674ef98fc83b7604be2ffc

Shahane, S. (2021). *ANT Detection Image Dataset* [Accessed: 2023-10-17].

Usuario 'AntsNet.org" en RoboFlow. (2021, septiembre). *Dataset 'AntsNet Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/antsnet/antsnet

Angulo, E., Hoffmann, B. D., Ballesteros-Mejia, L., Taheri, A., Balzani, P., Bang, A., Renault, D., Cordonnier, M., Bellard, C., Diagne, C., Ahmed, D. A., Watari, Y., & Courchamp, F. (2022). Economic costs of invasive alien ants worldwide. *Biological Invasions*, *24*(7), 2041-2060. https://link.springer.com/article/10.1007/s10530-022-02791-w

Jian, Z., Lu, Z., Zhou, X., Lan, B., Xiao, A., Wang, X., & Liang, B. (2022). PUTN: A Plane-fitting based Uneven Terrain Navigation Framework. *None*, 7.

Usuario 'ants" en RoboFlow. (2022, mayo). *Dataset 'ant_predict"* [Visitado el 2023-10-17]. https://universe.roboflow.com/ants/ant_predict

Usuario 'Personal" en RoboFlow. (2022, mayo). *Dataset 'Ants Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/personal-ryjmg/ants-2z73o

Wu, M., Cao, X., Yang, M., & amd Shihui Guo, X. C. (2022). A dataset of ant colonies' motion trajectories in indoor and outdoor scenes to study clustering behavior. *GigaScience*.

Baglioni, C. (2023). Leafy Grass Texture [Accessed: 2024-10-30]. https://polyhaven.com/a/leafy_grass

de Gier, D. (2023a, mayo). *Dataset 'Ant finder Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/djay-de-gier-rdrxp/ant-finder

de Gier, D. (2023b, septiembre). *Dataset 'Ant genus detections Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/djay-de-gier-k0ejr/ant-genus-detections

de Gier, D. (2023c, septiembre). *Dataset 'Ant object detection Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/djay-de-gier-fopbf/ant-object-detection

de Gier, D. (2023d, septiembre). *Dataset 'Ants classify Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/djay-de-gier-saqnt/ants-classify

de Gier, D. (2023e, septiembre). *Dataset 'Camponotus Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/djay-de-gier-xibjg/camponotus

de Gier, D. (2023f, septiembre). *Dataset 'New ants Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/djay-de-gier-2-uytbt/new-ants

de Gier, D. (2023g, octubre). *Dataset 'Pheidole Dataset" en RoboFlow* [Visitado el 2023-10-17]. https://universe.roboflow.com/djay-de-gier-xibjg/pheidole

de Gier, D. (2023h, septiembre). *Dataset 'Tapinoma Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/djay-de-gier-xibjg/tapinoma

Quixel. (2023). MegaScan Grass Clumps [Accessed: 2024-06-25]. https://quixel.com/megascans/home?search=grass&search=clumps&assetId=rbojr

Sabattini, J. A., Sturniolo, F., Bollazzi, M., & Bugnon, L. A. (2023). AntTracker: A low-cost and efficient computer vision approach to research leaf-cutter ants behavior. *ScienceDirect*.

Usuario 'intellaNest" en RoboFlow. (2023, agosto). *Dataset 'antDetect Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/intellanest/antdetect

Usuario 'Proyecto Hormiga" en RoboFlow. (2023, septiembre). *Dataset 'Ant Tracking Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/proyecto-hormiga/ant-tracking

Usuario 'TFG" en RoboFlow. (2023, agosto). *Dataset 'Ant Detection Dataset"* [Visitado el 2023-10-17]. https://universe.roboflow.com/tfg-gh7jg/ant-detection-gfju4

Amazon. (2024). Amazon. https://www.amazon.es/ODROID-N2-4GB-2-4GHz/dp/B09RQLZFZ2

Baglioni, C. (2024). Dirt Texture [Accessed: 2024-10-30]. https://polyhaven.com/a/dirt

Berois, M., de Oliveira, L., & Gastelú, E. (2024, septiembre). *GitLab Proyecto*. https://gitlab.fing.edu.uy/eduardo.gastelu/pg_robot_hormiguero

Blender. (2024). Blender. https://www.blender.org/

Doxygen. (2024, agosto). OpenCV [Visitado el 2024-08-13]. https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

Engine, U. (2024). Unreal Engine 5. https://www.unrealengine.com/en-US/unreal-engine-5

Meena, M. S., & Saini, L. S. (2024). Smart Pest Management: The Role of Precision Agriculture in Modern Farming. *AgriTech Revolution: Advancing Sustainable Farming Volume I*, 978-81. https://www.researchgate.net/publication/385565841_SMART_PEST_MANAGEMENT_THE_ROLE_OF_PRECISION_AGRICULTURE_IN_MODERN_FARMING

Robotis. (2024). AX-12A [Visitado el 2024-08-02]. https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

Robotis. (s.f.). *Comprehensive Kit Robot Series Manual* [Visitado el 2024-06-24].

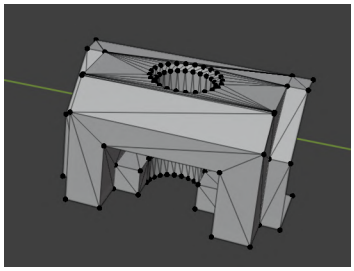
Anexo A

Anexo

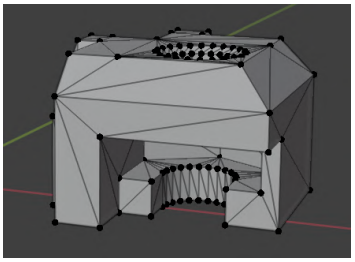
A.1. Modelado del robot

Modelado La creación de este modelo fue lograda utilizando el programa de modelado 3D Blender Blender, 2024, y un modelo 3D para referencia de la pieza original antes mencionada. Primero se cargó la pieza de referencia en Blender, la cuál no tenía una topología homogénea que nos permitiera modificarla directamente, por lo que se debió separar el modelo en distintas partes que nos permitan a partir de ellas generar un modelo de mayor tamaño que mantenga la forma adecuada para acoplarse a las demás piezas del kit y los motores.

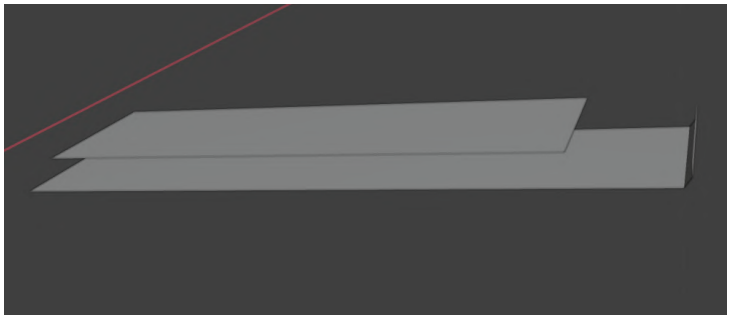
Estas piezas fueron remodeladas para tener una topología consistente por si solas y que les permite acoplarse con las demás, además de mantener la forma y tamaño necesarios para que los tornillos y enganches sigan funcionando al imprimirlos, como si fueran las originales. Fue necesario hacer piezas de la esquina izquierda, la esquina derecha - la cual es simétrica a la esquina izquierda, por lo que debió espejarse, ya que no se puede lograr con una rotación - y el borde. Además se crearon de cero cuatro segmentos, dos para la pared interna en el costado y la esquina y otros dos para las mismas en la pared externa, que se unen con facilidad a las piezas anteriores y mantienen la altura de las paredes del robot original, lo que permite conectar al nuevo robot a las patas originales. Estas se pueden ver en la figura A.1.



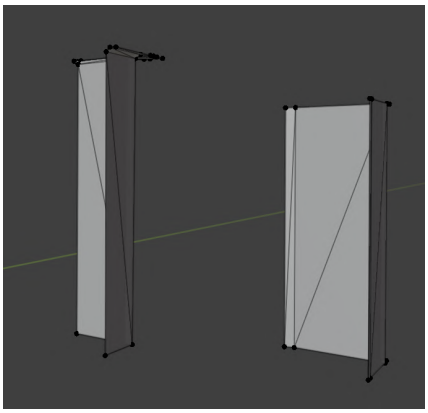
(a) Borde con tornillo



(b) Esquina del borde con tornillo



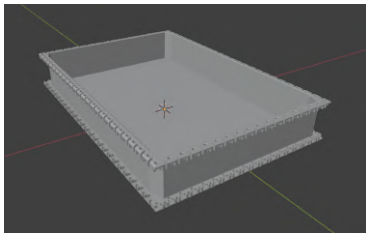
(c) Paredes laterales y frontales
Interna y externa



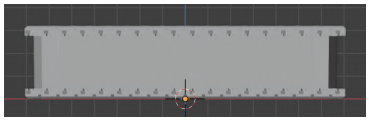
(d) Paredes de la esquina
Interna y externa

Figura A.1: Modelos de las piezas modulares diseñadas

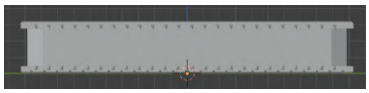
Una vez modeladas todas las piezas estas son conectadas y repetidas hasta obtener la distancia deseada para formar un cuarto de las paredes del robot, a la que se le agrega el piso con un plano de las dimensiones requeridas, y el modelo resultante es espejado en los dos ejes horizontales con respecto al origen - donde se encontraría el centro del modelo - para formar los otros tres cuadrantes faltantes, con el fin de luego unir todos los cuadrantes y obtener el modelo final. En la figura A.2 se encuentran las imágenes de los diseños creados.



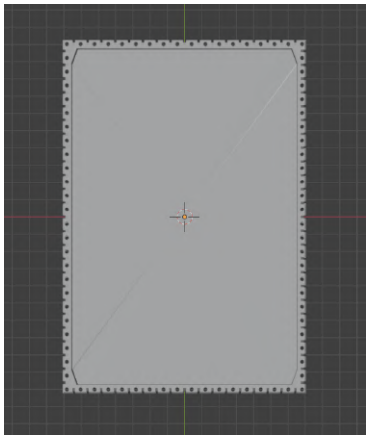
(a) En ángulo



(b) Frontal



(c) Lateral



(d) Superior

Figura A.2: Imágenes del nuevo modelo diseñado

Experimento Patas				
Ambiente	**Número de corrida**	**Desplazamiento frontal ↑**	**Desplazamiento lateral ↓**	**Desviación ↓**
Alfombra	1	80 cm	16 cm	11.3099°
Alfombra	2	85 cm	11 cm	7.3738°
Alfombra	3	86 cm	9 cm	5.9743°
Alfombra	4	86 cm	10 cm	6.6325°
Alfombra	5	88 cm	11 cm	7.1250°
Pasto Corto	1	78 cm	13 cm	9.4623°
Pasto Corto	2	81 cm	10 cm	7.0379°
Pasto Corto	3	81 cm	7 cm	4.9392°
Pasto Corto	4	81 cm	9 cm	6.3402°
Pasto Corto	5	81 cm	12 cm	8.4270°
Arena	1	114 cm	31 cm	15.2126°
Arena	2	108 cm	31 cm	16.0154°
Arena	3	114 cm	36 cm	17.5256°
Arena	4	118 cm	14 cm	6.7662°
Arena	5	114 cm	27 cm	13.3245°
Pasto Largo	1	45 cm	2 cm	2.5448°
Pasto Largo	2	91 cm	22 cm	13.5909°
Pasto Largo	3	78 cm	3 cm	2.2026°
Pasto Largo	4	70 cm	4 cm	3.2705°
Pasto Largo	5	40 cm	2 cm	2.8624°

Tabla A.1: Tabla de resultados del experimento sobre las patas seleccionadas en distintos entornos.

Resultados - Velocidades y Tiempo de Espera				
Velocidad	**Time.sleep (s)**	**Desplazamiento frontal ↑ (cm)**	**Desplazamiento lateral ↓ (cm)**	**Desviación (º)**
200	0.5 s	80 cm	16 cm	11.3099º
200	0.5 s	85 cm	11 cm	7.3737º
200	0.5 s	86 cm	9 cm	5.9743º
200	0.5 s	86 cm	10 cm	6.6325º
200	0.5 s	88 cm	11 cm	7.1250º
300	0.5 s	109 cm	54 cm	26.3544º
300	0.5 s	107 cm	57 cm	28.0446º
300	0.5 s	104 cm	64 cm	31.6075º
300	0.5 s	114 cm	45 cm	21.5409º
300	0.5 s	107 cm	63 cm	30.4889º
200	0.3 s	180 cm	85 cm	25.2777º
200	0.3 s	184 cm	111 cm	31.1009º
200	0.3 s	190 cm	69 cm	19.9588º
200	0.3 s	201 cm	96 cm	25.5296º
200	0.3 s	139 cm	59 cm	22.9993º
300	0.3 s	202 cm	23 cm	6.4958º
300	0.3 s	194 cm	36 cm	10.5126º
300	0.3 s	197 cm	10 cm	2.9059º
300	0.3 s	200 cm	8 cm	2.2906º
300	0.3 s	200 cm	0 cm	0º
400	0.3 s	152 cm	98 cm	32.8113º
400	0.3 s	152 cm	98 cm	32.8113º
400	0.3 s	160 cm	98 cm	31.4874º
400	0.3 s	156 cm	100 cm	32.6609º
400	0.3 s	153 cm	103 cm	33.9485º

Tabla A.2: Tabla de resultados del experimento de velocidades y tiempos de espera.

Glosario

Filtro de Kalman Es un conjunto de ecuaciones matemáticas que proporciona una solución computacional eficiente (recursiva) del método de mínimos cuadrados. El filtro es muy potente en varios aspectos: admite estimaciones de estados pasados, presentes e incluso futuros, y puede hacerlo incluso cuando se desconoce la naturaleza precisa del sistema modelado. 9, 10

GPR GPR es el acrónimo de "Gaussian Process Regression" (Regresión de Proceso Gaussiano), es un método de aprendizaje automático supervisado para realizar regresión, es decir, predecir valores continuos a partir de datos de entrada. 14

Lucas-Kanade El método de Lucas-Kanade es un método diferencial para la estimación de flujo óptico desarrollado por Bruce D. Lucas y Takeo Kanade. El método asume que el flujo es continuo en el vecindario del pixel considerado y resuelve las ecuaciones básicas de flujo óptico para todos los pixeles en ese vecindario usando el criterio de "mínimos cuadrados". 12

NMPC El Control Predictivo no Lineal Basado en Modelos (NMPC) es una técnica avanzada de control de procesos utilizada para gestionar sistemas dinámicos complejos y no lineales. Esta metodología se basa en un modelo matemático no lineal del sistema, utilizando este modelo para predecir el comportamiento futuro y optimizar las acciones de control en tiempo real . 14

PF PF es el acrónimo de Particle Filter o filtro de partículas en español. Es una técnica de estimación bayesiana que utiliza un conjunto de muestras (partículas) para representar una distribución de probabilidad de un estado, generalmente utilizado en problemas de localización y estimación en robótica. El filtro de partículas estima la posición y orientación del robot basándose en la información de los sensores y en un mapa conocido del entorno. 57

PF-RRT PF-RRT se refiere a una combinación de dos técnicas populares en robótica y planificación de trayectorias: el filtro de partículas (PF) y el árbol de búsqueda aleatoria rápida (RRT). 14

RRT RRT es el acrónimo de Rapidly-exploring Random Tree. Es un algoritmo de planificación de movimiento que construye de manera incremental un árbol de búsqueda. Se utiliza para encontrar trayectorias en espacios de alta dimensionalidad, y es especialmente útil cuando se trata de entornos con obstáculos. El algoritmo RRT selecciona aleatoriamente puntos en el espacio y trata de conectarlos de manera eficiente, asegurándose de no chocar con obstáculos. 57

SLAM SLAM es el acrónimo de *Simultaneous Localization and Mapping*. Es un conjunto de algoritmos utilizados en robótica y realidad aumentada que tiene dos objetivos principales: localización, es decir, determinar la posición y orientación de un robot o dispositivo en un entorno desconocido y mapeo, es decir, construir o actualizar un mapa de un entorno mientras se navega por él. 14