UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Título del Proyecto

MEMORIA PRESENTADA POR:

Carlos Mira López
Nicolás Miró Mira
Vittorio Alessandro Esposito Ceballos

MODELADO Y CONTROL DE ROBOTS

GRADO EN INFORMÁTICA INDUSTRIAL Y ROBÓTICA

***Curso**: 2025/2026*

Resumen

Índice general

Índice de figuras

Índice de tablas

1 Introducción, objetivos y estructura del documento

El presente proyecto aborda el desafío de la navegación autónoma en robótica móvil mediante el uso de percepción visual. Se centra en el modelado y control del robot TurtleBot3 dentro de un entorno de simulación de alta fidelidad (IsaacSim/IsaacLab). El objetivo fundamental es desarrollar estrategias de control que permitan al robot alcanzar objetivos definidos utilizando cámaras RGB/RGB-D, explorando tanto algoritmos de navegación clásicos como técnicas avanzadas de Aprendizaje por Refuerzo (Reinforcement Learning), evaluando su desempeño en escenarios controlados.

1.1 Introducción

La robótica móvil ha experimentado una transformación radical en la última década, pasando de ser una curiosidad académica a convertirse en un pilar fundamental de la Industria 4.0 y la logística moderna. Los robots móviles autónomos (AMR) son hoy esenciales en tareas de transporte de materiales, inspección de infraestructuras y robótica de servicios. Sin embargo, para que estos sistemas sean verdaderamente autónomos , deben poseer la capacidad de percibir su entorno, interpretarlo y tomar decisiones de navegación seguras y eficientes en tiempo real.

Tradicionalmente, la navegación se ha resuelto mediante mapas estáticos y sensores láser (LiDAR). No obstante, la tendencia actual se dirige hacia el uso de percepción visual (cámaras RGB y de profundidad) y técnicas de inteligencia artificial, que permiten obtener una información semántica del entorno mucho mejor y a un menor coste de hardware.

Este proyecto surge de la necesidad de comprender y dominar las técnicas modernas de control robótico. Específicamente, se justifica en la transición tecnológica actual que busca sustituir o complementar la programación clásica con algoritmos de aprendizaje. El uso de herramientas de simulación fotorrealista y física precisa, como NVIDIA IsaacSim e IsaacLab, permite entrenar y validar estos sistemas en entornos virtuales seguros antes de su despliegue físico, reduciendo riesgos y costes (concepto conocido como *Sim-to-Real*).

Desde una perspectiva docente y académica, este trabajo permite la adquisición de competencias transversales en modelado cinemático, visión por computador, integración de sensores y, fundamentalmente, en la aplicación de Deep Reinforcement Learning (Deep RL), una técnica que está redefiniendo el estado del arte en el control de robots dinámicos.

1.2 Objetivo general y objetivos específicos

El **objetivo principal** de este proyecto es diseñar, implementar y validar un sistema de control para la navegación autónoma del robot móvil TurtleBot3 en un entorno simulado, capaz de localizar y alcanzar objetivos visuales utilizando información sensorial proveniente de cámaras a bordo y algoritmos de inteligencia artificial.

Para alcanzar esta meta, se han definido los siguientes objetivos específicos:

1. **Modelado y configuración del entorno de simulación:** Importar y configurar el modelo cinemático y visual del TurtleBot3 en el entorno IsaacSim/IsaacLab, integrando sensores virtuales de visión (cámara RGB/RGB-D) y configurando escenarios con metas visuales específicas.

2. **Desarrollo del sistema de percepción y navegación:** Implementar los algoritmos necesarios para procesar la información visual del robot y traducirla en comandos de velocidad (lineal y angular) que permitan la navegación hacia el objetivo.

3. **Implementación de agentes de Aprendizaje por Refuerzo (RL):** Diseñar y entrenar un agente basado en RL (explorando arquitecturas con redes neuronales convolucionales - CNNs) capaz de aprender la política de navegación óptima mediante la interacción prueba-error con el entorno simulado.

4. **Validación y análisis comparativo:** Evaluar el rendimiento del sistema en términos de tasa de éxito, tiempo de llegada y suavidad de la trayectoria, comparando la robustez de las soluciones frente a cambios en la posición del objetivo o del entorno.

5. **Documentación y análisis crítico:** Analizar la viabilidad de la transferencia de estos algoritmos a robots de servicio reales y documentar el proceso técnico para futuras investigaciones.

1.3 Estructura del documento

La presente memoria se organiza en cinco capítulos que detallan el desarrollo del proyecto desde su concepción teórica hasta la validación de resultados:

- **Capítulo 1: Introducción, objetivos y estructura.** Introduce el contexto del problema, justifica la elección del TurtleBot3 y las herramientas de simulación, y define las metas del proyecto.

- **Capítulo 2: Antecedentes y estado del arte.** Revisa los fundamentos teóricos de la robótica móvil diferencial, la percepción visual y el Aprendizaje por Refuerzo. Se analizan trabajos previos y se describen las herramientas utilizadas (ROS2, IsaacSim, PyTorch).

- **Capítulo 3: Materiales y métodos.** Detalla la metodología empleada, incluyendo el modelado matemático del robot, la configuración de los sensores en simulación, la arquitectura de las redes neuronales utilizadas y el diseño de la función de recompensa para el entrenamiento del agente.

- **Capítulo 4: Resultados.** Presenta las gráficas de entrenamiento, las métricas de desempeño obtenidas en las pruebas de navegación y una comparativa de los distintos controladores implementados. Se valida el cumplimiento de los objetivos.

- **Capítulo 5: Conclusiones y trabajo futuro.** Sintetiza los hallazgos principales, discute las limitaciones encontradas durante el desarrollo y propone líneas de investigación para continuar el trabajo, como la implementación en hardware real.

2 Antecedentes y estado del arte

En este capítulo se debe contextualizar el proyecto dentro del conocimiento existente. No se trata de hacer un simple resumen, sino de demostrar que se entiende qué se ha hecho ya en el área, qué problemas siguen abiertos y qué aporta vuestro trabajo en ese contexto.

2.1 Fundamentos teóricos relevantes

2.2 Cinemática de Robots: La Geometría del Movimiento

La cinemática es la rama de la mecánica que describe el movimiento de los puntos, cuerpos (objetos) y sistemas de cuerpos (grupos de objetos) sin considerar las fuerzas que causan el movimiento. En robótica, el estudio de la cinemática se centra en la relación geométrica entre las articulaciones del robot (espacio articular) y la posición y orientación de su efector final (espacio de tareas).

2.2.1 Cinemática Directa y la Convención Denavit-Hartenberg (DH)

La Cinemática Directa (FK, por sus siglas en inglés) resuelve la pregunta: *"Dadas las posiciones de todas las articulaciones, ¿dónde está la mano del robot?"*. Para un manipulador serial, que consiste en una cadena de eslabones rígidos conectados por articulaciones, la posición del efector final se calcula mediante la composición de transformaciones homogéneas sucesivas.

Para estandarizar este proceso y evitar ambigüedades en la definición de los sistemas de coordenadas locales de cada eslabón, se emplea universalmente la convención de parámetros de Denavit-Hartenberg (DH). Este método reduce la descripción de la geometría espacial de cualquier mecanismo serial a cuatro parámetros fundamentales por eslabón:

- **Longitud del eslabón (a_i):** Distancia a lo largo del eje x_i (la normal común) entre los ejes z_{i-1} y z_i.
- **Torsión del eslabón (α_i):** Ángulo entre los ejes z_{i-1} y z_i medido alrededor del eje x_i.

- **Desplazamiento del eslabón (d_i):** Distancia a lo largo del eje z_{i-1} desde el origen del sistema de coordenadas $i-1$ hasta la intersección con el eje x_i. En articulaciones prismáticas, esta es la variable.

- **Ángulo de la articulación (θ_i):** Ángulo entre los ejes x_{i-1} y x_i medido alrededor del eje z_{i-1}. En articulaciones rotativas, esta es la variable.

La matriz de transformación homogénea ${}^{i-1}T_i$ que describe la posición y orientación del sistema de referencia i con respecto al sistema $i-1$ se construye matemáticamente como:

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

El modelo cinemático total del robot, que relaciona la base (sistema 0) con el efector final (sistema n), es el producto matricial de estas transformaciones individuales:

$$ {}^{0}T_n = \prod_{i=1}^{n} {}^{i-1}T_i $$

Esta formulación matricial es crucial porque permite a los controladores computar la posición cartesiana en tiempo real con un costo computacional determinista y bajo.

2.2.2 Cinemática Inversa (IK): El Problema Mal Planteado

La Cinemática Inversa (IK) aborda el problema opuesto y mucho más complejo: *"Dada una posición y orientación deseada del efector final, ¿qué valores deben tener las articulaciones (q)?"*. A diferencia de la FK, la IK no siempre tiene una solución única y cerrada.

- **Soluciones Múltiples:** Para un brazo robótico antropomórfico típico de 6 grados de libertad (DoF), pueden existir hasta 16 soluciones teóricas para una misma pose final (ej. configuraciones de çodo arriba"vs. çodo abajo").

- **Redundancia:** En robots con más de 6 DoF (robots redundantes), existen infinitas soluciones, lo que permite optimizar criterios secundarios como la evitación de obstáculos o la minimización de torques, pero complica enormemente la resolución matemática.

- **Singularidades:** Existen configuraciones donde el robot pierde grados de libertad efectivos. Matemáticamente, esto ocurre cuando el determinante de la matriz Jacobiana se anula.

Los métodos numéricos iterativos, como el método de Newton-Raphson o la optimización basada en gradientes (Gradiente Descendente, Levenberg-Marquardt), son estándares en la robótica moderna para resolver la IK en tiempo real.

2.2.3 La Matriz Jacobiana: Velocidad y Estática

La matriz Jacobiana $J(q)$ es, quizás, la herramienta matemática más importante en el control de manipuladores. No solo relaciona las velocidades, sino que conecta dominios físicos dispares.

Mapeo de Velocidades

Relaciona la velocidad articular $\dot{q}$ con la velocidad cartesiana del efector final v:

$$v = \begin{bmatrix} v_{lineal} \\ \omega_{angular} \end{bmatrix} = J(q)\dot{q} \tag{2.2}$$

Esto permite controlar el movimiento suave del robot en el espacio cartesiano ajustando las velocidades de los motores.

Mapeo de Fuerzas (Estática)

A través del principio del trabajo virtual, la Jacobiana transpuesta relaciona los torques en las articulaciones τ con las fuerzas y momentos F aplicados en el efector final:

$$\tau = J^T(q)F \tag{2.3}$$

Esta relación es fundamental para el control de impedancia y cumplimiento, permitiendo que un robot "sienta"fuerzas externas o aplique fuerzas precisas sin sensores de fuerza dedicados en cada articulación, basándose en la corriente de los motores.

2.3 Dinámica de Robots: Fuerzas y Energía

Mientras que la cinemática trata la geometría, la dinámica estudia las fuerzas necesarias para causar dichas aceleraciones. Un modelado dinámico preciso es esencial para el control de alta velocidad y para las simulaciones realistas necesarias en el aprendizaje por refuerzo.

2.3.1 Formulación Lagrangiana

El enfoque Lagrangiano se basa en el balance de energía del sistema. Se define el Lagrangiano $\mathcal{L}$ como la diferencia entre la energía cinética total $\mathcal{K}$ y la energía potencial total $\mathcal{P}$ del sistema ($\mathcal{L} = \mathcal{K} - \mathcal{P}$). Aplicando las ecuaciones de Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \tag{2.4}$$

Se obtiene la ecuación dinámica cerrada estándar en robótica:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + F_{fric}(\dot{q}) = \tau \tag{2.5}$$

Donde cada término tiene una interpretación física crítica para el control:

- $M(q)$ **(Matriz de Inercia):** Es simétrica y definida positiva. Representa la resistencia del robot a acelerar. A diferencia de una masa escalar constante, $M(q)$ cambia con la configuración del robot.
- $C(q,\dot{q})\dot{q}$ **(Fuerzas de Coriolis y Centrípetas):** Representan fuerzas ficticias que surgen en sistemas de referencia rotatorios. Los términos centrípetos dependen de $\dot{q}_i^2$, mientras que los de Coriolis dependen del producto $\dot{q}_i\dot{q}_j$.
- $g(q)$ **(Vector de Gravedad):** El torque necesario solo para mantener el robot estático contra la gravedad.

2.3.2 Formulación Newton-Euler Recursiva (RNEA)

Aunque la formulación Lagrangiana es elegante analíticamente, es computacionalmente costosa ($O(n^4)$ o $O(n^3)$). Para la simulación y el control en tiempo real, se prefiere el algoritmo Newton-Euler Recursivo (RNEA), que tiene una complejidad lineal $O(n)$. RNEA funciona en dos pasadas:

1. **Pasada hacia adelante (Forward Pass):** Calcula velocidades y aceleraciones de cada eslabón desde la base hasta el efector final.
2. **Pasada hacia atrás (Backward Pass):** Calcula las fuerzas y torques necesarios para crear esas aceleraciones, propagándolas desde el efector final hacia la base.

2.4 Teoría de Control Clásico

El control clásico se basa en modelos matemáticos explícitos para garantizar la estabilidad y el seguimiento de trayectorias.

2.4.1 Control PID (Proporcional-Integral-Derivativo)

El PID sigue siendo el caballo de batalla de la industria. Calcula la señal de control $u(t)$ basándose en el error $e(t) = r(t) - y(t)$:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{2.6}$$

- **Proporcional (K_p):** Respuesta inmediata.
- **Integral (K_i):** Elimina el error de estado estacionario.
- **Derivativo (K_d):** Predice el error futuro (amortiguamiento).

Limitaciones en Robótica: El PID asume sistemas lineales e invariantes en el tiempo (LTI). Dado que los robots son altamente no lineales, a menudo se requiere "Gain Scheduling.º términos de "Feedforward"dinámico.

2.4.2 Control Predictivo Basado en Modelos (MPC)

El MPC utiliza un modelo dinámico del robot para predecir su comportamiento futuro en un horizonte de tiempo finito H y optimizar las acciones de control. En cada paso de tiempo t, resuelve:

$$\min_{u_{t:t+H}} \sum_{k=0}^{H} \left(||x_{t+k} - x_{ref}||_Q^2 + ||u_{t+k}||_R^2 \right) \tag{2.7}$$

Sujeto a restricciones como la dinámica del robot ($x_{k+1} = f(x_k, u_k)$), límites de actuadores y evitación de colisiones. El MPC maneja explícitamente las restricciones físicas, siendo ideal para sistemas inestables como bípedos.

2.5 Fundamentos de Aprendizaje por Refuerzo (RL)

Cuando los modelos analíticos son insuficientes, el RL ofrece un marco para aprender el control a partir de la experiencia.

2.5.1 Procesos de Decisión de Markov (MDP)

El problema se formaliza como una tupla (S, A, P, R, γ): Estado (S), Acción (A), Transición $(P(s'|s,a))$ y Recompensa $(R(s,a))$.

2.5.2 La Ecuación de Bellman

La base de la mayoría de los algoritmos de RL es la Ecuación de Bellman:

$$V(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right) \tag{2.8}$$

Esta recursividad permite propagar recompensas futuras hacia atrás en el tiempo. En Deep RL, $V(s)$ o $Q(s,a)$ se aproximan mediante redes neuronales profundas.

2.6 Estado del arte en robótica y control

El estado del arte (SOTA) en robótica entre 2024 y 2025 se define por una transición acelerada desde sistemas rígidos basados en modelos analíticos hacia sistemas adaptativos basados en datos ("Data-Driven").

2.6.1 La Transición de Model-Based a Learning-Based

Tradicionalmente, se asumía que un modelo físico preciso (Matriz $M(q)$, $C(q,\dot{q})$) permitía un control perfecto. Sin embargo, dinámicas no modeladas (fricción, holguras) y la interacción con entornos no estructurados hacen que el Deep Reinforcement Learning (DRL) sea la solución dominante para aprender políticas robustas $\pi(a|s)$.

2.6.2 Algoritmos de Deep RL Dominantes en Robótica

- **Proximal Policy Optimization (PPO):** El estándar para locomoción (cuadrúpedos, humanoides). Es un método *On-Policy* que utiliza una función objetivo recortada"para evitar actualizaciones catastróficas.
- **Soft Actor-Critic (SAC):** Preferido para manipulación y robots reales. Es *Off-Policy* y maximiza la entropía:

$$J(\pi) = \sum_t \mathbb{E}[r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))]$$

- **DDPG y TD3:** Gradualmente desplazados por SAC, aunque útiles en vehículos autónomos (políticas deterministas).

Característica	PPO	SAC	DDPG / TD3
Tipo	On-Policy	Off-Policy	Off-Policy
Eficiencia Muestras	Baja	Alta	Media/Alta
Estabilidad	Muy Alta	Alta (Entropía)	Media
Aplicación	Locomoción	Manipulación, Real	Vehículos

Tabla 2.1: Comparativa de algoritmos de RL

2.6.3 El Desafío Sim-to-Real

Para cerrar la "Brecha de Realidad", se utilizan técnicas avanzadas:

- **Domain Randomization (DR) y ADR:** Aleatorización de parámetros físicos y visuales para que el mundo real sea solo üna variación más".
- **Adaptación Online (RMA):** Una red neuronal comprime el historial de observaciones en un vector latente que permite a la política adaptarse en tiempo real.
- **Maestro-Estudiante:** Una política "Maestroçon información privilegiada entrena a una política .Estudiante"que solo usa sensores reales.

2.6.4 Arquitecturas Híbridas y Casos de Estudio

- **Residual RL:** Combina un controlador clásico (u_{base}) con una corrección de RL ($u_{residual}$), ideal para ensamblajes de contacto rico.
- **Manipulación Móvil Armónica:** Control de cuerpo completo (Whole-Body Control) mediante RL para coordinar base y brazo.
- **Modelos VLA:** Integración de LLMs para razonamiento semántico (recoge el animal extinto") junto con controladores de bajo nivel.

2.7 Herramientas y entornos de simulación

2.7.1 Motores de Física y Fidelidad

NVIDIA Isaac Sim y PhysX 5

Estándar industrial (2024-2025). Utiliza Ray Tracing (RTX) para alta fidelidad visual y permite **Paralelismo Masivo**: A través de Isaac Lab (antes Orbit), simula miles de robots en una sola GPU, reduciendo tiempos de entrenamiento drásticamente.

MuJoCo

Estándar académico. Destaca por su estabilidad matemática en cadenas complejas. **MJX (2024 Update):** Permite ejecutar la física de MuJoCo directamente en TPUs y GPUs usando JAX, igualando velocidades con Isaac Sim.

Gazebo y PyBullet

Gazebo sigue siendo vital para la integración con ROS, mientras que PyBullet está siendo relegado al prototipado rápido.

2.7.2 Comparativa Técnica de Simuladores

Característica	Isaac Sim / Lab	MuJoCo / MJX	Gazebo	PyBullet
Motor Física	PhysX 5 (GPU)	General Coords	DART/ODE	Bullet (CPU)
Enfoque	RL Masivo, Visión	Investigación	ROS, Sistema	Prototipado
Paralelismo	Extremo (GPU)	Extremo (JAX)	Bajo	Medio
Fidelidad Visual	Muy Alta (RTX)	Media	Media	Baja
Sim-to-Real	Excelente	Excelente	Bueno	Moderado

Tabla 2.2: Comparativa de Simuladores en 2025

2.7.3 Tendencias Emergentes

- **Simulación Diferenciable (Brax, Dojo):** Permite propagar gradientes a través del motor físico para optimización analítica.
- **Rendering Neuronal (NeRF/Gaussian Splatting):** Reconstrucción 3D basada en IA para crear entornos de simulación fotorrealistas a partir de escaneos del mundo real.

2.8 Conclusiones y Perspectivas Futuras

La robótica actual combina los fundamentos teóricos ineludibles (cinemática y dinámica) con nuevas capas de decisión basadas en Aprendizaje por Refuerzo y Simulación Masiva. El futuro apunta a la convergencia de simulación fotorrealista y modelos fundacionales multimodales, permitiendo a los robots no solo moverse, sino comprender su entorno.

2.8.1 Principios Básicos del Aprendizaje por Refuerzo

Breve repaso de los conceptos esenciales de cinemática y dinámica de robots.

Introducción a los esquemas de control clásicos (PID, control por realimentación, control óptimo, etc.).

Principios básicos del aprendizaje por refuerzo (Markov Decision Processes, política, recompensa, exploración vs. explotación).

2.9 Estado del arte en robótica y control

Aplicaciones recientes de cinemática y dinámica en robots manipuladores y móviles.

Técnicas actuales de control: comparación entre enfoques tradicionales y métodos basados en inteligencia artificial.

Uso del aprendizaje por refuerzo en robótica: principales algoritmos (DQN, PPO, SAC, DDPG) y retos en su implementación.

2.10 Herramientas y entornos de simulación

Breve revisión de los simuladores más utilizados en robótica (Gazebo, MuJoCo, PyBullet, IsaacSim).

Justificación del uso de IsaacLab/IsaacSim para este proyecto.

2.11 Trabajos y resultados previos

Ejemplos de investigaciones o proyectos similares que hayan aplicado RL en robótica.

Limitaciones encontradas en esos trabajos y vacíos de investigación que motivan este proyecto.

2.12 Relación con el proyecto

Identificación de qué aspectos se tomarán como base para el trabajo (ej. modelado cinemático/dinámico tradicional).

Qué parte supone un reto o innovación (ej. implementación de RL en IsaacLab).

3 Materiales y métodos

La sección Materials and Methods (también llamada Methodology o Experimental Section, según la disciplina) es una parte esencial de artículos y memorias de ámbito académico/docente. Su objetivo principal es que otro estudiante, profesor, investigador, ingeniero, etc., pueda reproducir el trabajo siguiendo las descripciones dadas.

3.1 Modelado cinemático/dinámico del robot + tool (si aplica)

Descripción de los pasos seguidos para modelar el robot dentro del entorno de simulación. Si se ha realizado un modelado teórico, incorporar: Formulación de la cinemática directa e inversa mediante el uso de matrices de transformación homogénea, parámetros de Denavit–Hartenberg. Obtención de la matriz Jacobiana y análisis de singularidades.

Modelado dinámico (ecuaciones de Euler–Lagrange o Newton–Euler) con las hipótesis adoptadas (ej. robot rígido, sin rozamiento, etc.).

Representación clara de ecuaciones y, cuando sea útil, diagramas que apoyen la comprensión.

3.2 Metodología de control y aprendizaje por refuerzo (RL)

Explicación de los esquemas de control diseñados: control clásico de referencia, control basado en RL, comparación, etc.

3.2.1 Esquema de control

Control por LIDAR y cámara

Este control se basa en el guiado por el entorno basandose en al información recibida por el LIDAR, basandose en los datos frontales y laterales que recibimos. Con la cámara lo que hacemos es buscar el color que tiene el objetivo, en nuestro caso es el color rojo, por lo que se va analizando la imagen constantemente, ya que si detectamos un área mínima del

color del objetivo, extraemos cuanto estamos descentrados para poder centrarse y poder ir recto al objetivo.

Control por RL

Justificación de la elección de algoritmos de RL (ej.: PPO, SAC, DDPG), con breve descripción de su funcionamiento.

3.2.2 Algoritmo RL PPO

Definición de recompensas, estados y acciones empleados en el entorno de simulación.

3.2.3 Recompensas, estados y acciones

recompensas

Las recompensas que se le han asignado al robot se basán en si se ha descubierto area o no dentro de la imagen de la cámara. Se han definido zonas de distancia de LIDAR al robot para darle más recompensa o menos:

- Distancia frontal libre(FRONT_CLEAR) : cuando el lidar por delante detecta más de 10m
- Distancia frontal peligrosa(FRONT_DANGER) : Cuando la distancia frontal del LIDAR por delante baja de los 7m
- Distancia muy peligrosa : Es la distancia antes del choque donde antes de chocarse prefiero que vaya hacia atrás el robot, 5.5m

Primero se comprueban las recompensas por alcanzar el objetivo, o por colisión:

- Si se alcanza el objetivo : +30.0
- Si se ha colisionado : −8.0

Si no se ha alcanzado el área mínima de 1000px, las recompensas son las siguientes:

- Si delante es muy libre, es decir, por encima del umbral FRONT_CLEAR, y el robot tiene la acción de ir hacia delante rapido o lento le damos 0.3 por rápido, y 0.15 por lento
- Si delante esta por debajo del umbral FRONT_DANGER y el robot tiene la accion de ir hacia delante, le quitamos 0.6

- Si delante está por debajo del FRONT_CLEAR, y giramos hacia el lado que más depejado esta, le damos 0.4, con esto lo que hacemos es que si por delante vemos una distancia donde no cabe el robot, es inecesario entrar dentro de esa sala, es mejor girar

Estados

Acciones

- Moverse hacia delante rápido
- Moverse hacia delante lento
- Moverse fuerte hacia la derecha
- Moverse suave hacia la derecha
- Moverse fuerte hacia la izquierda
- Moverse suave hacia la izquierda
- Moverle lento hacia antrás

3.3 Entorno de simulación y herramientas utilizadas

Descripción de IsaacSim/IsaacLab: versiones empleadas, configuración inicial y librerías auxiliares. Recursos computacionales (hardware, GPU, sistema operativo).

Configuración de escenarios de entrenamiento (robot, entorno, sensores virtuales, condiciones de interacción).

3.3.1 Unity + Visual estudio

3.3.2 Visual estudio code

3.3.3 TensorFlow

TensorFlow Dashboard es una herramienta que a través de los logs que genera nuestro entrenamiento, va graficando cada 4024 pasos, que son 256 por robot, se grafican los resultados.

3.3.4 KERAS + gym

3.3.5 ROS 1

Hemos empleado ROS1 como pasarela de comunicación entre los robots y el máster, que sería nuestro propio ordenador. Lo que hacemos es que cada robot tiene unos tópics propios, sobre los cuales envían y reciben la información.

3.3.6 Turtle bot

3.3.7 Entorno

3.3.8 Sensores virtuales

LIDAR

Hemos modelado un LIDAR que publica los valores en un topic de ros, es este colisionado es un codigo de Unity el cual genera una nube de puntos, y los publica todos a través del tópic

3.3.9 Condiciones de iteraccion

3.4 Procedimiento de experimentación / entrenamiento

Explicación del flujo de trabajo seguido: preparación de modelos, definición de hiperparámetros, duración de entrenamientos, validación de políticas aprendidas.

3.4.1 Flujo de trabajo

Se probo primero definiendo un escenario muy sencillo, donde si el robot giraba en el primer cruze ya alcanzaba el objetivo final. Esto se modificó y se hicieron 4 escenarios para el mismo entrenamiento, donde el objetivo estaba en diferentes puntos.

Estrategia de comparación: métricas definidas (tiempo de convergencia, estabilidad, error en el seguimiento, etc.).

3.4.2 Estrategia de comparacion

Se definió que cada 1000 pasos en cada robot, se realize una comparcion de modelo, pero se detendria el entrenamineto si la media de los ultimos 1000 episodios no mejoraba durante los 5 siguientes episodios.

Número de episodios, pruebas o repeticiones realizadas.

3.5 Programación

Diagramas de flujo o pseudocódigo de los programas implementados.

4 Resultados

En este capítulo se deben presentar y analizar los resultados obtenidos tras la implementación del proyecto. No se trata solo de mostrar datos, gráficos o tablas, sino de interpretarlos y relacionarlos con los objetivos planteados.

4.1 Validación del modelado matemático

Comparación entre los resultados teóricos (cinemática y dinámica calculadas) y los obtenidos en simulación.

Comprobación de trayectorias, posiciones finales y detección de posibles singularidades.

Discusión sobre la precisión y las limitaciones del modelo.

4.2 Entrenamientos en IsaacLab/IsaacSim

Presentación de las curvas de aprendizaje (recompensa acumulada, convergencia del algoritmo).

Evaluación de los tiempos de entrenamiento y de la estabilidad de la política aprendida.

Análisis de hiperparámetros: cómo afectan al rendimiento y a la velocidad de convergencia.

4.3 Comparación de controladores

Resultados de control clásico vs. control basado en RL.

Métricas de desempeño: error en el seguimiento de trayectorias, suavidad de los movimientos, robustez frente a perturbaciones.

Discusión sobre ventajas y limitaciones de cada enfoque.

4.4 Pruebas finales y validación del sistema

Ejemplos de simulaciones finales con el robot ejecutando las tareas propuestas.

Posibles animaciones, capturas de pantalla o gráficas que ilustren los comportamientos logrados.

Evaluación cualitativa (ej.: naturalidad de movimientos, respuesta a cambios en el entorno).

4.5 Discusión crítica

Relación de los resultados con los objetivos generales y específicos planteados en la introducción.

Identificación de fortalezas, limitaciones y posibles mejoras del trabajo.

5 Conclusiones y trabajo futuro

En este capítulo tendrás la oportunidad de realizar las conclusiones de tu proyecto, volviendo a señalar los aspectos más importantes que se puede ver en la memória, los logros conseguidos, etc. Además, deberás exponer aquellos aspectos en los que piensas que se podría trabajar de cara a ofrecer una mejor solución que la propuesta y/o ampliar la solución.

5.1 Conclusiones

5.2 Trabajo futuro