UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Control de un robot por Vision y LIDAR

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. El trabajo se centra en el modelado y control del robot TurtleBot3 dentro de un entorno de simulación desarrollado en ***Unity****. El objetivo fundamental es implementar estrategias de control que permitan al robot alcanzar objetivos definidos utilizando cámaras RGB/RGB-D, explorando tanto algoritmos de navegación clásicos como técnicas de Aprendizaje por Refuerzo (Reinforcement Learning), y evaluando su desempeño en escenarios controlados.*

1.1 Introducción

La robótica móvil ha evolucionado significativamente en los últimos años, consolidándose como un elemento clave en la Industria 4.0 y la logística. Los robots móviles autónomos (AMR) son esenciales para tareas de transporte, inspección y servicios. Sin embargo, para que estos sistemas operen de forma efectiva, requieren la capacidad de percibir su entorno y tomar decisiones de navegación en tiempo real. Tradicionalmente, la navegación se ha basado en mapas estáticos y sensores láser (LiDAR). No obstante, existe una tendencia creciente hacia el uso de percepción visual y técnicas de inteligencia artificial, lo cual permite obtener información semántica del entorno reduciendo la dependencia de sensores costosos.

Este proyecto surge de la necesidad de aplicar técnicas modernas de control robótico. Se justifica en el interés actual por complementar la programación clásica con algoritmos de aprendizaje. Para ello, se opta por el uso de Unity como motor de simulación. Unity ofrece un entorno físico robusto y flexible que permite entrenar y validar estos sistemas de forma segura antes de una eventual implementación física, reduciendo riesgos y costes asociados a las pruebas en hardware real.

Desde una perspectiva académica, este trabajo facilita la adquisición de competencias en modelado cinemático, visión por computador y la aplicación de Deep Reinforcement Learning (Deep RL), permitiendo estudiar el comportamiento de agentes autónomos en entornos virtuales complejos.

1.2 Objetivo general y objetivos específicos

El **objetivo principal** es diseñar, implementar y validar un sistema de control para la navegación autónoma del TurtleBot3 en un entorno simulado en Unity, capaz de localizar y alcanzar objetivos visuales utilizando información de cámaras a bordo.

Para alcanzar esta meta, se definen los siguientes objetivos específicos:

1. **Configuración del entorno de simulación:** Integrar el modelo cinemático y visual del TurtleBot3 en el motor Unity, incorporando sensores virtuales de visión y laser(LiDAR) y diseñando escenarios de prueba con metas específicas.

2. **Desarrollo del sistema de navegación:** Implementar los algoritmos de control necesarios para procesar la información visual y generar los comandos de velocidad (lineal y angular) para el desplazamiento del robot.

3. **Validación y análisis:** Evaluar el rendimiento del sistema midiendo la tasa de éxito y la eficiencia de las trayectorias, comparando el comportamiento del robot ante diferentes configuraciones del entorno.

4. **Documentación:** Registrar el proceso técnico, los problemas encontrados y las soluciones adoptadas, analizando la viabilidad de los algoritmos propuestos.

1.3 Estructura del documento

La memoria se organiza en cinco capítulos que cubren desde la teoría hasta los resultados experimentales:

- **Capítulo 1: Introducción, objetivos y estructura.** Presenta el contexto del problema, justifica el uso de Unity y el TurtleBot3, y define las metas del trabajo.

- **Capítulo 2: Antecedentes y estado del arte.** Revisa los fundamentos de la robótica diferencial y el Aprendizaje por Refuerzo. Se describen las herramientas clave utilizadas.

- **Capítulo 3: Materiales y métodos.** Detalla la metodología, incluyendo el modelado del robot en Unity, la configuración de los sensores, la arquitectura de control y el diseño de la función de recompensa.

- **Capítulo 4: Resultados.** Muestra las métricas de desempeño obtenidas en las pruebas de navegación, gráficas de entrenamiento y una discusión sobre el comportamiento del robot.

- **Capítulo 5: Conclusiones y trabajo futuro.** Sintetiza los logros del proyecto, discute las limitaciones detectadas y propone posibles mejoras o líneas de continuación.

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

2.1.1 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).

2.1.2 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.

2.1.3 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.

2.1.4 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.

2.2 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.

2.2.1 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.

2.2.2 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.

2.3 Teoría de Control Clásico

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

2.3.1 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.

2.3.2 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.

2.4 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.

2.4.1 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 $(R(s,a))$.

2.4.2 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.

2.5 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").

2.5.1 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 $\pi(a|s)$.

2.5.2 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

2.5.3 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.

2.5.4 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.

2.6 Herramientas y entornos de simulación

2.6.1 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.

2.6.2 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

2.6.3 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.

2.7 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 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.

2.9 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 Dinámico

El objetivo de esta sección es obtener las ecuaciones que gobiernan el movimiento del robot JetBot 3 relacionando los pares aplicados por los motores con las aceleraciones del sistema. Para ello, se utiliza el formalismo de **Euler-Lagrange**.

3.1.1 Hipótesis Adoptadas

Para la derivación del modelo matemático, se asumen las siguientes simplificaciones sobre la física del robot:

- **Cuerpo Rígido:** El chasis y las ruedas son indeformables.
- **Movimiento Plano:** El robot se desplaza sobre una superficie horizontal, por lo que la energía potencial gravitatoria es constante ($V = 0$).
- **Rodadura Pura:** Se asume la condición de no deslizamiento longitudinal ni lateral en las ruedas (restricción no holonómica).
- **Simetría:** Se asume que el centro de masa (CoM) del robot se encuentra en el punto medio del eje que une las dos ruedas motrices.

3.1.2 Definición de Coordenadas

La postura del robot en el entorno global se define mediante el vector de coordenadas generalizadas q:

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3.1}$$

Donde (x, y) representan la posición cartesiana del punto medio del eje de las ruedas y θ la orientación del chasis respecto al eje X global.

Figura 3.1: Esquema cinemático del robot diferencial (JetBot) con las variables de estado y parámetros geométricos.

3.1.3 Formulación de Euler-Lagrange

La función Lagrangiana $\mathcal{L}$ se define como la diferencia entre la energía cinética (T) y la energía potencial (V) del sistema:

$$\mathcal{L}(q, \dot{q}) = T(q, \dot{q}) - V(q) \tag{3.2}$$

Dado que el movimiento es plano ($V = 0$), el Lagrangiano es igual a la energía cinética total, compuesta por la traslación lineal y la rotación angular:

$$\mathcal{L} = \frac{1}{2} m(\dot{x}^2 + \dot{y}^2) + \frac{1}{2} I_z \dot{\theta}^2 \tag{3.3}$$

Siendo m la masa total del robot e I_z el momento de inercia respecto al eje vertical que pasa por el centro de masas.

Las ecuaciones de movimiento se obtienen aplicando la ecuación de Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = B(q)\tau - A^T(q)\lambda \tag{3.4}$$

Donde τ son las fuerzas generalizadas (pares de los motores), $B(q)$ es la matriz de transformación de entrada y $A^T(q)\lambda$ representa las fuerzas de restricción (fricción de rodadura lateral) que limitan el movimiento a la dirección de las ruedas.

3.1.4 Modelo Dinámico Resultante

Resolviendo las derivadas y considerando la relación cinemática entre la velocidad del robot y la velocidad angular de las ruedas $(\dot{\phi}_R, \dot{\phi}_L)$, se llega a la forma compacta del modelo dinámico en el espacio de estados:

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} + \mathbf{F}_{fric}(\dot{q}) = \frac{1}{R} \begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ L/2 & -L/2 \end{bmatrix} \begin{bmatrix} \tau_R \\ \tau_L \end{bmatrix} \tag{3.5}$$

Donde:

- R: Radio de las ruedas.
- L: Distancia entre las ruedas (ancho de vía).
- τ_R, τ_L: Pares aplicados por los motores derecho e izquierdo.

3.2 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.

3.3 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.

3.3.1 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.

3.3.2 Algoritmo RL PPO

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

3.3.3 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

3.4 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).

3.4.1 Unity + Visual estudio

3.4.2 Visual estudio code

3.4.3 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.

3.4.4 KERAS + gym

3.4.5 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.

3.4.6 Turtle bot

Aunque en nuestros resultados podemos observar que el robot es un cubo amarillo, lo que se esta haciendo es simular el comportamiento del turtle bot en un cubo con sus ecuaciones dinámicas

3.4.7 Entorno

Nuestro entorno es un laberito con paredes y pasillos por donde el robot deberá moverse y encontrar el objetivo de color rojo que puede estar en cualquier lado de la escena. Se diseñó un mismo entorno, donde en cada instancia el objetivo estaba en una posición diferente como vemos en las siguientes imagenes.

3.4.8 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. En nuestro código lo tratamos de manera que lo separamos en 180 sectores, esto lo hacemos cogiendo el minimo cada dos sectores.

3.4.9 Condiciones de iteraccion

3.5 Procedimiento de experimentación / entrenamiento

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

3.5.1 Flujo de trabajo

Se probó 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.).

3.5.2 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.

3.6 Programación

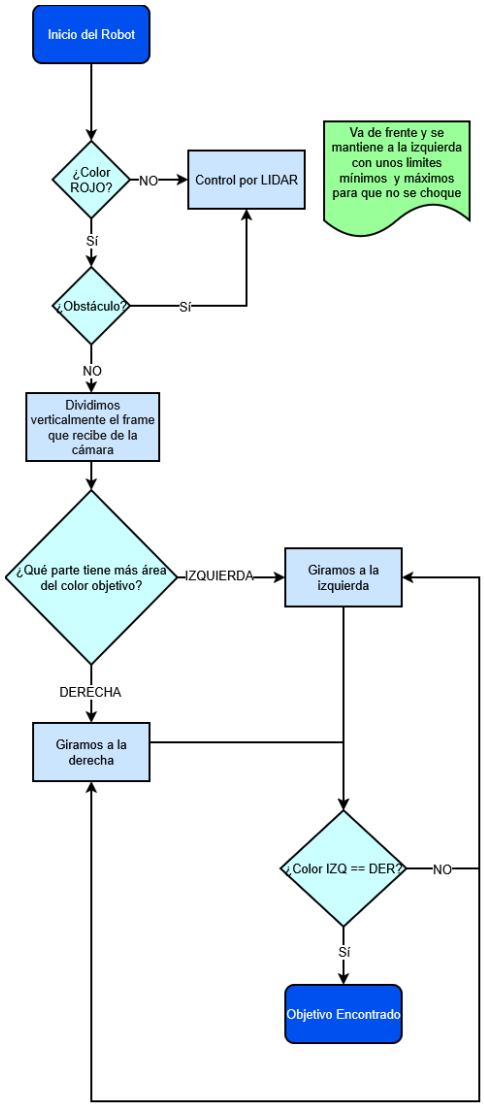

Figura 3.2: Diagrama de Flujo del control por LIDAR y cámara

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 Entrenamientos en Unity

Entrenamiento 1 en reinforce learning

Vamos a analizar las gráficas obtenidas por el entrenamiento que podemos observar más adelante (4.3.1). Podemos observar en la siguiente imagen (4.1) que conforme van haciendo más pasos, la longitud por episodio es mayor lo que significa que cada vez investiga más el modelo, y a su vez se puede observar que la recompensa media por episodio va aumentando en cada paso, lo que nos indica que cada vez va haciendo pasos más correctos y chocandose menos.

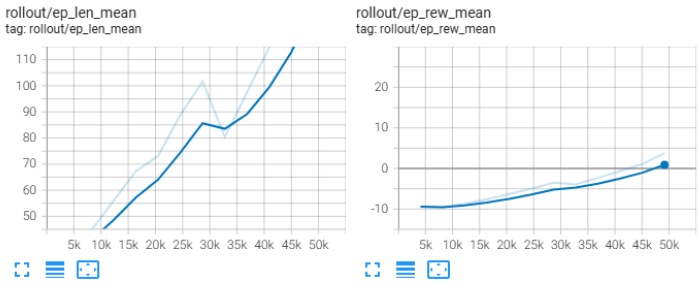

Figura 4.1: Longitud media y recompensa media por episodio

Como observamos en la siguiente gráfica, vemos que cada vez la "Entropy loss " es menor, lo que significa que el modelo cada vez toma acciones menos aleatorias, como también nos indica la grafica de la varianza, donde vemos que cada vez es mayor, y el modelo cada vez entiende más el resultado, aunque no lo alcanze, ya que los valores recomdados son entre $0.8, 0.9$ para que el modelo entienda mejor.

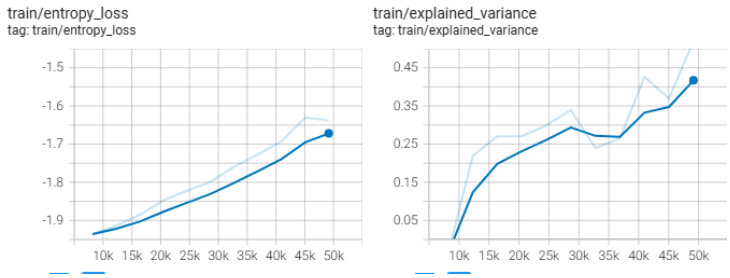

Figura 4.2: Longitud media y recompensa media por episodio

Con esto podemos pensar que el modelo comprende cada vez mejor el entorno y lo que tiene que realiar, aunque como veremos más adelante, este no lo consigue, puede ser falta de entrenamiento, ya que no se llego a detener ni por pasos máximos, ni por early stopping, sino porque llegadas casi las 2h de tener el ordenador bloqueado se tuvo que detener y coger un checkpoint que se van generado.

Entrenamiento 2 en reinforce learning

En este entrenamiento nos fijamos mirando las gráficas que a pesar de que la recompensa aumenta, la varianza deja de explicar el modelo a los 24000 pasos, y al realizar la prueba con ese modelo, a pesar de que el early stopping salta a los 42000 pasos , parece que es mejor modelo el de los 24000 que es el que se ha probado en el test que vemos más adelante. (4.3.1)

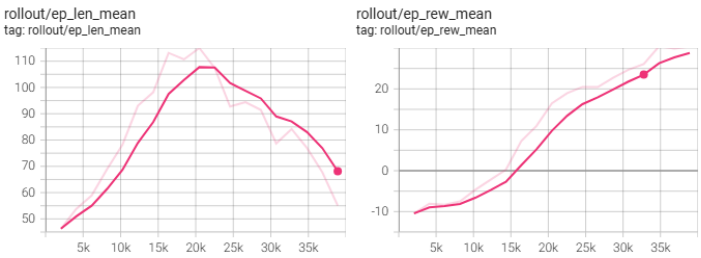

Figura 4.3: Longitud media y recompensa media por episodio

¡

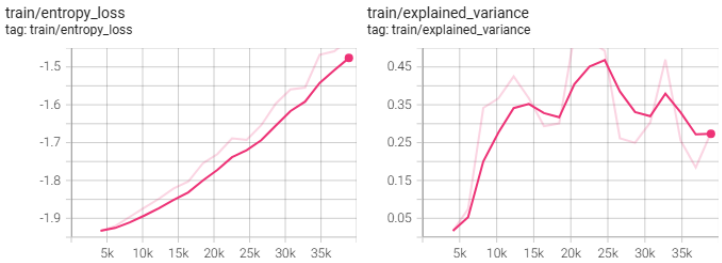

Figura 4.4: Longitud media y recompensa media por episodio

4.2 Comparación de controladores

4.2.1 Comparación control clásico vs control basado en RL

-NO ME HA DADO TIEMPO A EDITAR EL VIDEO- Como observamos en el siguiente video : podemos observar en una pantalla el comportamiento del robot en cuatro escenarios moviendose con el LIDAR, mientras que en la derecha vemos el robot intentando alcanzar un objetivo después de realizar un entrenamiento. Podemos observar que con el LIDAR no produce colisiones, ya que es muy estricto con toda la información del LIDAR que tiene, manteniendo la distancia con la pared izquierda sin problemas, mientras que con el comportamiento en RL vemos que se queda atascado en un giro, donde no mantiene la distancia con el lIDAR y se colisona.

4.2.2 Ventajas y limitaciones

Las ventajas del control por LIDAR, es que resuelve el escenario correctamente, ya que siempre irá apegado a la pared izquierda, pero esto tiene un gran inconveniente y esque podria llegar a escanear todo el entorno si el objetivo esta a la derecha y oculto, esto a difencia del RL es un problema, ya que con el RL se intenta aprender un patrón para no tener que ir siempre apegado a la izquierda y poder hacer los movimientos más centrados.

4.3 Pruebas finales y validación del sistema

4.3.1 Pruebas finales

Aqui dejo el enlace al video donde se ve a 4 robots resolviendo 4 escenarios donde el objetivo esta en diferentes posiciones: ENLACE.

Ahora dejamos varias pruebas que se han realizado para el reinforce learning con diferentes recompensas, os ponemos el video del entrenamiento con la grafica de recompensa media cada 4048 pasos de evaluación donde hay que tener en cuenta que no se han podido completar los entrenamientos por duración, y se han detenido cuando llevavan aproximadamente 1h, aunque hay alguno que dura 2h, y otro que se detiene por early stopping.

- Entrenamiento 1 16 instancias.
- Test 1 16 instancias.
- Entrenamiento 2 8 instancias.
- Test 2 8 instancias.

4.4 Discusión crítica

Relación de los resultados con los objetivos generales y específicos planteados en la introducción. Respecto a los objetivos que propusimos en la introducción, hemos conseguido alcanzar todos los específicos, ya que estos eran el control de un robot con camara y sensorica por un entorno para alcanzar el objetivo, esto se ha conseguido ya que como hemos visto anteriormente, el robot consigue alcanzar el objetivo con la cámara y el LIDAR, mientras que respecto a los específicos, hemos intentado el del reinforce learning, consiguiendo entrenar el robot, y que el robot se desplazara por el laberinto aprendiendo la politica del robot con el LIDAR de seguir la paret izquierda, pero se colisionaba alfinal sin conseguir alcanzar el objetivo final.

4.4.1 Fortalezas, limitaciones y mejoras

Las fortalezas que tiene el proyecto, es que el control por cámara y LIDAR es muy robusto, porque tiene la politica muy bien establecida de ir apegado a la izquierda y recto si no hay obtáculo, la límitiación que aparece es que hay caminos que muy probablemente se los deje sin investigar o que tarde mucho en alcanzarlos ya que tiene que recorrer todo el recorrido.

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

La realización de este proyecto ha permitido validar la integración efectiva de un entorno de simulación basado en Unity y ROS, demostrando ser una plataforma viable para el prototipado de robótica móvil. Los resultados experimentales han arrojado una clara distinción entre los enfoques evaluados: el control clásico, basado en reglas de percepción láser y visual, se consolidó como la solución más robusta, logrando completar los objetivos de navegación sin colisiones gracias a la estricta gestión de los umbrales de seguridad definidos.

Por el contrario, la implementación del agente de Aprendizaje por Refuerzo mediante el algoritmo PPO evidenció las dificultades inherentes al entrenamiento de redes neuronales en entornos complejos. Aunque el agente logró inferir una política de navegación básica, no consiguió la convergencia necesaria para garantizar la seguridad del robot, resultando en choques frecuentes y bloqueos en geometrías difíciles. Esto confirma que, si bien el RL ofrece una gran flexibilidad teórica, su implementación práctica requiere un ajuste de hiperparámetros y un diseño de recompensas mucho más exigente que la programación algorítmica tradicional.

5.2 Trabajo futuro

Para superar las limitaciones actuales, la prioridad será migrar el entorno de entrenamiento a NVIDIA Isaac Sim e Isaac Lab. Esta herramienta permitirá aprovechar la simulación masiva en paralelo en GPU, reduciendo drásticamente los tiempos de cálculo. Además, se explorarán algoritmos más eficientes como SAC para mejorar la convergencia, con el objetivo final de transferir el modelo aprendido al robot físico.