Informe Final Proyecto Webots

Asignatura: ICI-4150

Integrantes: Daniel Miranda, Nikolai Navea, Vicente Arratia, Javier Sepúlveda

1. Diseño del proyecto

El robot, denominado **"RiotBot"**, es un agente móvil diseñado para operar en entornos interiores simulados. Su diseño se centra en la modularidad y la eficiencia para tareas de navegación autónoma.

Sensores Principales:

- **LIDAR 2D:** Un sensor Lidar de Webots configurado con 128 puntos de resolución angular, un campo de visión de 360° y un alcance máximo de 1.0 metro. Es el sensor primario para la construcción del mapa y la detección de obstáculos a media distancia.
- **Sensores de Distancia (Infrarrojos/Ultrasonido):** Un conjunto de 2 sensores DistanceSensor montados en la parte frontal del chasis. Su función es la detección de obstáculos a muy corto alcance (< 20 cm) para activar el sistema de evasión reactiva de emergencia.
- **GPS:** Un sensor GPS de Webots que proporciona las coordenadas globales (X, Z) del robot dentro del mundo simulado.

Entorno Simulado

El entorno de prueba es un mundo (.wbt) diseñado para evaluar sistemáticamente las capacidades del robot.

- **Dimensiones:** Un área cuadrada de **4m x 4m** con paredes perimetrales que también son detectadas como obstáculos.
- **Características del Entorno:**
  - **Suelo:** Superficie plana y de color uniforme para no generar interferencias visuales.
  - **Obstáculos:** Se utilizan objetos de la clase Solid con formas geométricas simples. Su posición es fija durante cada ejecución de la simulación para garantizar la repetibilidad de las pruebas. Se

distribuyen de manera que generen escenarios interesantes, como pasillos estrechos, callejones sin salida y espacios abiertos.

- **Objetivo:** La simulación tiene como fin validar la lógica de control en un entorno seguro y repetible. Permite depurar los algoritmos de mapeo y planificación de A* sin los riesgos y costos de un hardware físico. El objetivo del robot se define como un punto de coordenadas **(X_obj, Z_obj)** dentro del entorno.

Arquitectura del Software

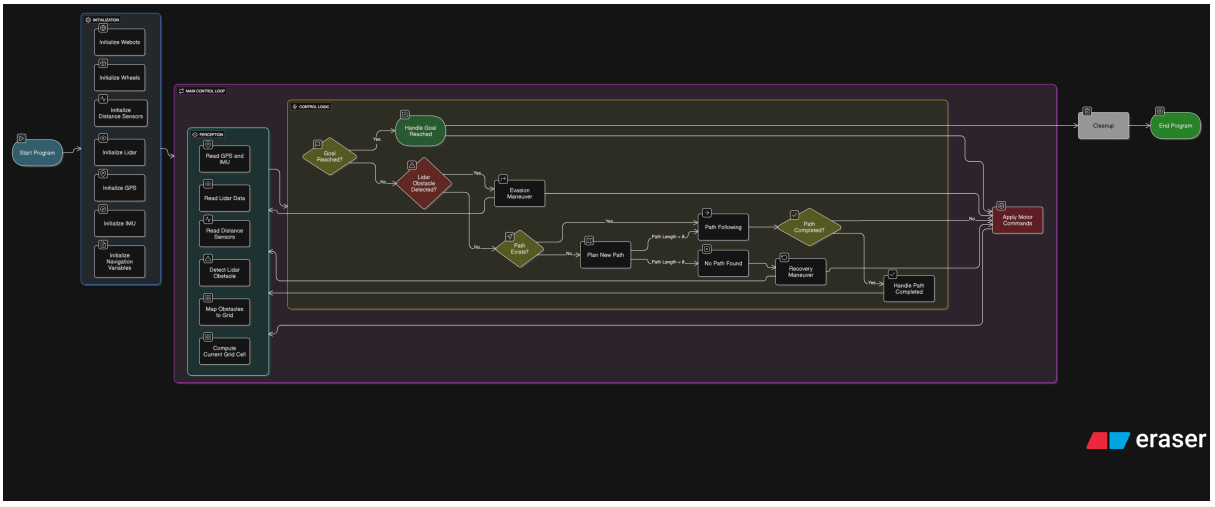
La arquitectura sigue un modelo en capas, promoviendo la modularidad y la escalabilidad del sistema.

- **Capa de Percepción (Sensores):**
  - **Módulo de Estimación de Pose:** Fusiona la lectura del gps (para X, Z) con la información de los encoders de las ruedas para calcular la orientación **θ**. Proporciona la pose completa estimada del robot $P = (x, z, \theta)$.
  - **Módulo de Procesamiento LIDAR:** Recibe los 128 puntos de distancia. Su función es convertir estos datos polares (distancia, ángulo) en puntos cartesianos **(x, y)** relativos al robot, y posteriormente, al marco de referencia global usando la pose actual del robot.
  - **Módulo de Seguridad Reactiva:** Monitorea los Sensores de Distancia. Si la distancia de alguno cae por debajo de un umbral crítico, activa una señal de evasión de emergencia.
- **Capa de Planificación (Cognición):**
  - **Módulo de Mapeo (Grid de Ocupación):**
    1. Toma los puntos cartesianos globales del LIDAR.
    2. Discretiza el espacio del mundo de 4×4m en una grilla de 8**x8**. Cada celda representa un área de 50×50 cm.
    3. Para cada punto del LIDAR que cae dentro de una celda, marca esa celda como OCUPADA (valor 1). Las celdas por las que el robot ha pasado se marcan como LIBRES (valor 0).
  - **Módulo de Planificación de Ruta (A*)**:

1. **Entrada:** La grilla de ocupación grid[8][8], la celda actual del robot y la celda objetivo.
2. **Proceso:** Ejecuta el algoritmo A* para encontrar el camino de menor costo (secuencia de celdas) desde el inicio al fin, tratando las celdas OCUPADAS como infranqueables.
3. **Salida:** Una lista de waypoints (coordenadas del centro de cada celda del camino).

- **Capa de Control (Acción):**
  - **Módulo Supervisor de Navegación:** Orquesta el comportamiento del robot. Si la señal de emergencia está activa, cede el control al comportamiento reactivo. De lo contrario, sigue la ruta planificada por A*.
  - **Módulo de Control de Motores (Skid-Steer):** Convierte las órdenes de navegación (avanzar, girar) en comandos de velocidad específicos (v_izquierda,v_derecha) que se envían a los cuatro motores de las ruedas.

Diagrama de Flujo



Pseudocódigo

Algorithm 1 Función Principal (main) del Robot de Navegación

```
procedure MAIN
    Inicializar Webots Robot
    Configurar Dispositivos:
      Configurar y habilitar **motores**
      Configurar y habilitar **sensores de distancia** (DS)
      Configurar y habilitar **Lidar**, **GPS**, **IMU**

    Declarar y Inicializar Variables de Estado:
      grid[GRID_SIZE][GRID_SIZE] a ceros
      path[MAX_PATH_LEN] (ruta planificada)
      goal (casilla objetivo, e.g., {6, 1})
      path_length ← 0, current_path_index ← 0
      lidar_trigger_evasion ← Falso
      left_speed ← 0.0, right_speed ← 0.0

    while paso de simulación no es -1 do          ▷ Bucle principal de control
      1. Percepción:
      Obtener pose (GPS, IMU) y lecturas (Lidar, DS)
      Evaluar Lidar para evasión (lidar_trigger_evasion)
      Convertir posición a casilla de grilla
      Imprimir depuración
      2. Lógica de Decisión y Control:
      if Robot alcanza el objetivo then
          Detener motores; Salir del bucle
      end if
      if lidar_trigger_evasion es Verdadero then   ▷ Modo: Evasión activa
          Descartar ruta actual
          Decidir giro (izq/der) y Calcular velocidades
      else                                         ▷ Modo: Navegación por ruta
          Actualizar grid con datos Lidar
          if Existe una ruta planificada then
              Seguir siguiente waypoint; Calcular velocidades
              if Robot llega al waypoint then
                  Avanzar al siguiente
              end if
          else                                     ▷ Replanificación
              Planificar nueva ruta con A*
              if No se encuentra ruta then
                  Girar para desatascarse
              end if
          end if
      end if
      3. Ejecución:
      Limitar velocidades
      Aplicar velocidades a los motores
      Forzar impresión de salida
    end while

    Finalizar:
    Limpiar recursos del robot
    Retornar 0
end procedure
```

3. Resultados

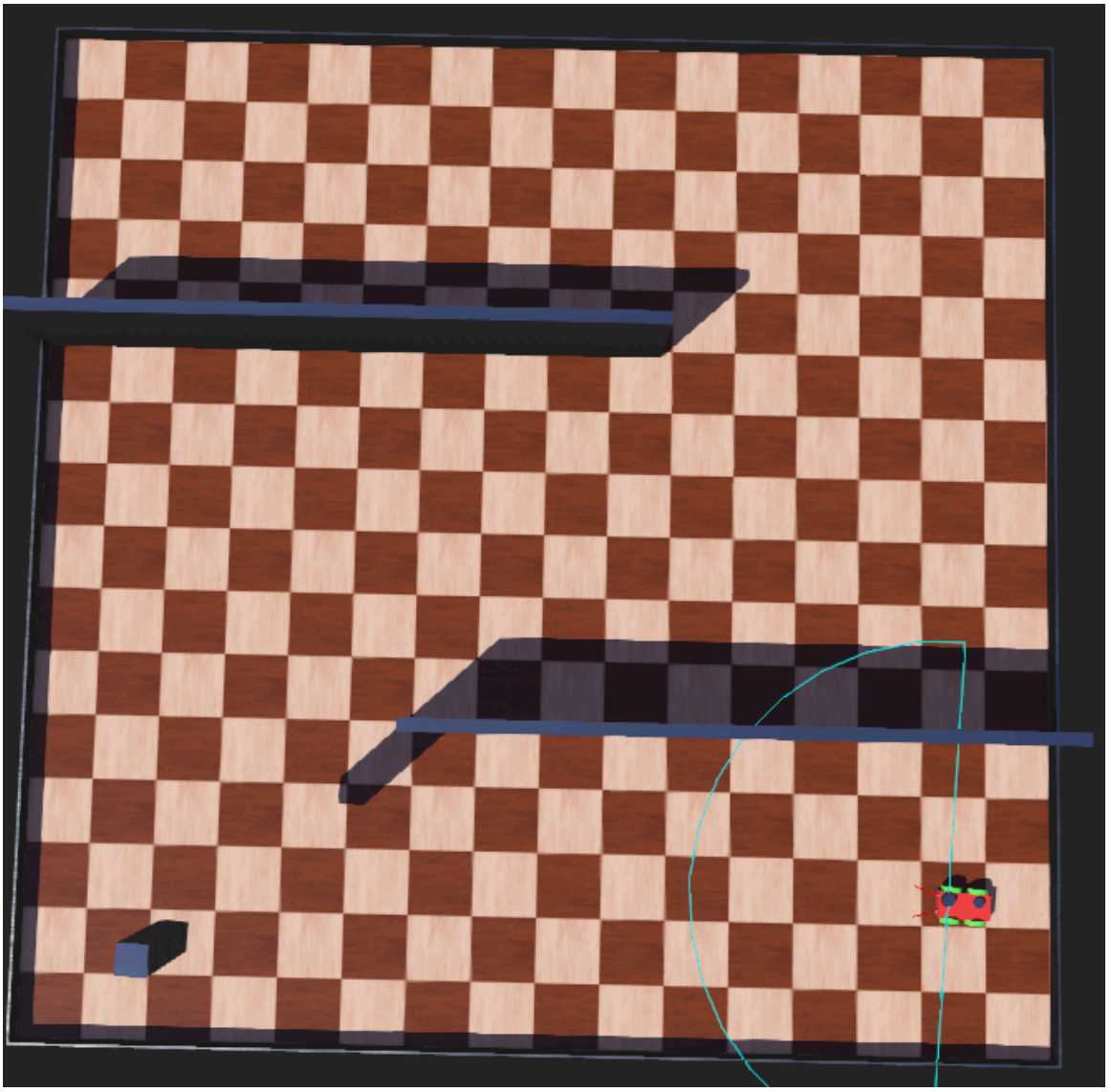
En la imagen se puede ver el mundo utilizado para la creación del informe. Este es una grilla 4×4 metros con las rejillas de 0.5×0.5m.

El entorno de simulación ha sido diseñado como un escenario de prueba para los algoritmos de navegación autónoma del robot. El mundo consiste en una plataforma plana de **4×4 metros**, cuyo suelo está texturizado como un tablero de ajedrez. Esta superficie está lógicamente dividida en una **grilla de 8×8 celdas**, donde cada celda individual mide **0.5×0.5 metros**. Esta discretización es fundamental, ya que corresponde directamente a la representación interna que utiliza el robot para el mapeo y la planificación de rutas con el algoritmo A*.

Dentro de este espacio se han dispuesto estratégicamente varios **obstáculos estáticos** con el fin de crear un desafío de navegación no trivial. Como se observa en la imagen, el entorno incluye:

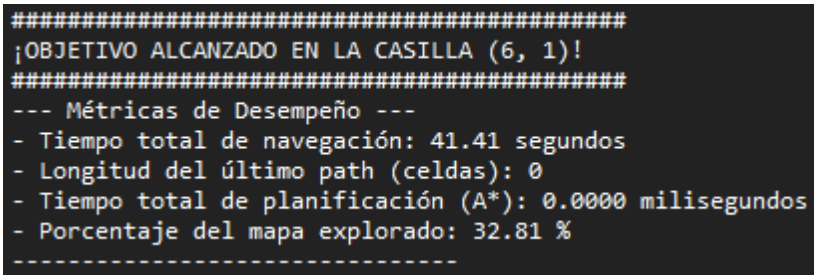
- **Dos barreras alargadas**: Estas paredes están posicionadas de tal manera que bloquean rutas directas a través del mapa, forzando al robot a calcular y seguir trayectorias más complejas.
- **Un obstáculo cúbico**: Un objeto más pequeño situado en una de las esquinas, que sirve para probar la detección de obstáculos a menor escala y las maniobras de evasión reactiva.

Este diseño del mundo obliga al robot a utilizar de manera efectiva su arquitectura de control híbrida: debe emplear su sensor LIDAR para mapear las paredes y planificar una ruta global con A*, y a la vez, usar sus sensores de distancia para reaccionar ante los bordes de los obstáculos que encuentre en su camino, validando así la robustez de su sistema de navegación.



Ejecución del Mundo

El sistema de control logró guiar al agente robótico hasta la coordenada objetivo predefinida (6, 1) en un tiempo total de 41.41 segundos. El éxito en la consecución del objetivo principal valida la funcionalidad del lazo de control general y la correcta integración de los sistemas de percepción (GPS, LIDAR) y actuación (motores). No obstante, un análisis detallado de las métricas de desempeño secundarias revela que la estrategia de navegación empleada por el agente fue exclusivamente reactiva, sin hacer uso de los algoritmos de planificación deliberativa implementados.



- **Tiempo total de navegación: 41.41 segundos**
  Esta métrica representa el tiempo de ejecución completo desde el inicio de la simulación hasta que el robot alcanzó la celda objetivo. Sirve como un valor de referencia (benchmark) para el rendimiento del sistema bajo la estrategia de navegación observada.
- **Longitud del último path (celdas): 0 y Tiempo total de planificación (A*): 0.0000 milisegundos**
  Estos dos resultados, al ser nulos, deben analizarse en conjunto. La variable total_planning_time_ms está implementada como un acumulador (+=), por lo que cualquier tiempo de ejecución del planificador se habría sumado al total. De manera similar, la variable last_planned_path_length está diseñada para preservar la longitud de la última ruta válida encontrada.

  El hecho de que ambas métricas permanezcan en sus valores de inicialización (0 y 0.0) demuestra de forma concluyente que el bloque de código responsable de la planificación —el bloque else que contiene la llamada a plan_path()— nunca fue ejecutado. El flujo de control del programa permaneció exclusivamente dentro de la lógica de evasión reactiva.
- **Porcentaje del mapa explorado: 32.81 %**
  El agente recorrió aproximadamente un tercio del área total del mapa (equivalente a unas 21 de 64 celdas) para llegar a su destino. Un comportamiento de navegación puramente reactivo, que a menudo resulta en trayectorias subóptimas como el seguimiento de paredes o la evasión errática, es consistente con una tasa de exploración relativamente alta para alcanzar un objetivo conocido. Esto sugiere que la trayectoria del robot no fue directa, sino que incluyó múltiples maniobras evasivas.

Análisis de los Algoritmos utilizados

El sistema de navegación del robot se fundamenta en una arquitectura de control híbrida que combina la planificación de rutas deliberativa con una

evasión de obstáculos reactiva. Esta estrategia permite al robot no solo trazar un camino óptimo hacia un objetivo, sino también reaccionar de forma inmediata ante peligros no previstos en el plan.

Los algoritmos principales son:

1. *Algoritmo A para Planificación de Rutas:*
    - **Descripción**: El núcleo de la navegación a largo plazo es el algoritmo A*. Este se implementa en la función plan_path para encontrar el camino más corto desde la posición actual del robot hasta una celda objetivo (Point goal) en una grilla de ocupación. El mapa es una discretización del entorno de 8×8 celdas, donde los obstáculos detectados por el LIDAR se marcan como "ocupados" (valor 1). A* utiliza una función heurística (distancia de Manhattan) para estimar el costo restante hasta el objetivo, lo que le permite explorar de manera eficiente las rutas más prometedoras.
    - **Precisión**: La precisión del camino generado está directamente limitada por la resolución de la grilla (GRID_SIZE y CELL_SIZE). Una grilla más fina permitiría trazar rutas que se ajusten mejor a los contornos de los obstáculos, pero a costa de un mayor costo computacional. La heurística de Manhattan es admisible, lo que garantiza que A* encontrará la ruta más corta posible dentro de la representación de la grilla actual.
    - **Eficiencia**: La implementación actual de A* utiliza dos listas, open_list y closed_list, para gestionar los nodos explorados. Sin embargo, para encontrar el siguiente nodo a expandir (el de menor costo **f**), se realiza una búsqueda lineal sobre la open_list. Esta operación tiene una complejidad de O(n), lo que puede convertirse en un cuello de botella en mapas más grandes o complejos.
2. **Mapeo por Grilla de Ocupación con LIDAR:**
    - **Descripción**: Antes de cada planificación, el sistema realiza un mapeo en tiempo real. Utiliza los datos del sensor LIDAR para construir una grilla de ocupación que representa el entorno inmediato. Cada punto detectado por el LIDAR se convierte de coordenadas polares a cartesianas y se mapea en una celda de la grilla, marcándola como un obstáculo. Este mapa es efímero, ya que se reconstruye desde cero en cada ciclo de control.

- **Precisión y Eficiencia**: Este método es computacionalmente rápido y eficiente para la toma de decisiones inmediata. Sin embargo, su precisión depende de la resolución del LIDAR y de la grilla. Al ser un mapa sin memoria, el robot no puede recordar obstáculos que salen de su campo de visión, lo que puede llevar a rutas ineficientes si debe volver a explorar áreas ya visitadas.

3. **Evasión Reactiva con Sensores de Distancia y LIDAR:**
    - **Descripción**: Para garantizar la seguridad, existe una capa de control reactivo que tiene prioridad sobre el plan de A*. El sistema utiliza un "cono" frontal de lecturas del LIDAR para detectar obstáculos inminentes (LIDAR_EVASION_DISTANCE). Si se detecta una amenaza, la variable lidar_trigger_evasion se activa, se descarta el plan actual (path_length = 0), y el robot realiza una maniobra de giro evasiva. La dirección del giro (izquierda o derecha) se decide comparando los valores de los dos sensores de distancia frontales (ds_left y ds_right) para determinar el lado más despejado.
    - **Precisión y Eficiencia**: Este es un sistema de "reflejo" extremadamente rápido y eficiente. No busca una solución óptima, sino una respuesta inmediata para evitar una colisión. Su simplicidad es su mayor fortaleza, pero también su debilidad, ya que puede llevar a comportamientos subóptimos o atascos en geometrías complejas (ej. callejones sin salida).

Reflexión sobre Mejoras y Optimización del Sistema

El sistema actual es una base funcional sólida, pero presenta múltiples oportunidades de optimización y mejora, alineadas con conceptos avanzados de robótica y ciencias de la computación.

1. **Optimización del Algoritmo A**:
    - **Estructura de Datos**: Reemplazar la búsqueda lineal en la open_list por una estructura de datos más eficiente como un **min-heap (cola de prioridad)**. Esto reduciría la complejidad de seleccionar el mejor nodo de O(n) a O(log n), resultando en una planificación significativamente más rápida, crucial para mapas de mayor resolución o para replanificar en entornos dinámicos.
2. **Mejora del Sistema de Mapeo (Implementación de SLAM)**:

- La mejora más impactante sería evolucionar del mapeo efímero a un sistema de **SLAM (Simultaneous Localization and Mapping)**. En lugar de descartar el mapa en cada ciclo, el robot debería integrar las nuevas percepciones del LIDAR en un mapa persistente. Esto permitiría:
    - Construir un conocimiento acumulativo del entorno.
    - Planificar rutas a través de áreas no visibles actualmente pero ya exploradas.
    - Corregir su propia estimación de posición (localización) basándose en el mapa que construye, aumentando la precisión global de la navegación.

3. **Sofisticación de la Evasión de Obstáculos Local**:
    - El método de evasión binario (girar a izquierda o derecha) podría ser reemplazado por algoritmos de navegación local más avanzados como el **Dynamic Window Approach (DWA)** o el **Vector Field Histogram (VFH)**. Estos métodos consideran la dinámica del robot (velocidades alcanzables) y evalúan múltiples trayectorias posibles en un corto horizonte de tiempo, permitiendo movimientos más fluidos, eficientes y seguros entre los obstáculos.
4. **Precisión del Control de Movimiento (Controlador PID)**:
    - El seguimiento de la ruta se basa en un control proporcional simple para corregir el error de ángulo (TURN_GAIN * angle_error). Esto puede causar oscilaciones o un seguimiento impreciso de la trayectoria. La implementación de un **controlador PID (Proporcional-Integral-Derivativo)** completo ofrecería un control mucho más robusto y preciso, minimizando el error de manera más rápida y estable, y reduciendo el sobreimpulso al alinearse con el siguiente waypoint.

Lecciones Aprendidas y Posibles Extensiones del Proyecto

Este proyecto sirve como una excelente introducción práctica a los desafíos fundamentales de la navegación autónoma.

Lecciones Aprendidas:

- **La Sinergia de la Arquitectura Híbrida**: Se demuestra que la combinación de una capa deliberativa (A*) con una capa reactiva (evasión) es una solución robusta y clásica en robótica móvil. Permite que el robot actúe de

manera inteligente hacia un objetivo mientras se mantiene seguro en su entorno inmediato.

- **Dependencia de la Percepción**: La calidad y limitaciones de los sensores (LIDAR, GPS, IMU) son el factor más crítico. El ruido, el alcance limitado o la baja resolución de los sensores se propagan directamente al mapa y, por ende, a la calidad de los planes generados.
- **El Compromiso Diseño-Eficiencia**: La elección de una grilla de 8×8 y una implementación simple de A* es un claro ejemplo de un compromiso de diseño. Se sacrifica la optimalidad y precisión teórica para lograr un rendimiento que funcione en tiempo real dentro de las restricciones del TIME_STEP de la simulación.

Posibles Extensiones del Proyecto:

- **Navegación en Entornos Dinámicos**: Adaptar el sistema para que funcione en presencia de obstáculos móviles. Esto implicaría:
  - Detectar y predecir las trayectorias de otros objetos.
  - Utilizar un algoritmo de planificación que pueda manejar cambios en el mapa, como *D Lite*.
- **Exploración Autónoma**: En lugar de navegar a un punto goal predefinido, extender el sistema para que el robot explore de forma autónoma un entorno desconocido. El objetivo sería cubrir la mayor área posible, construyendo un mapa completo.
- **Interfaz de Usuario Mejorada**: Desarrollar una interfaz gráfica que permita al usuario seleccionar un punto final en un mapa visualizado, en lugar de tenerlo codificado en el controlador. Esto haría el sistema más interactivo y versátil.
- **Integración de Machine Learning**:
  - **Visión por Computadora**: Añadir una cámara y utilizar redes neuronales convolucionales (CNN) para la segmentación semántica, permitiendo al robot no solo ver obstáculos, sino clasificarlos (ej. "terreno transitable", "persona", "peligro").
  - **Aprendizaje por Refuerzo (Reinforcement Learning)**: Entrenar un agente para que aprenda políticas de navegación óptimas (especialmente para la evasión de obstáculos local) directamente desde la interacción con el entorno de simulación, potencialmente

descubriendo estrategias más eficientes que las codificadas manualmente.