Reporte Final

Alvaro Eduardo Lozano Medina, Luis Santiago Sauma Peñaloza,
Erick Segura Sánchez, Santiago De la Riva Juárez, Isaac Hernández Pérez
Equipo 4, TC2008B.302
Tecnológico de Monterrey,
Monterrey 64700, Mexico,
E-mails: {A00837585, A00836418 , A01613821, A01749140, A01198674}@tec.mx

***Resumen*—Una famosa compañía busca optimizar un espacio industrial abandonado mediante un sistema logístico automatizado por robots guiados por láser (LGVs) los cuales gestionan el almacenamiento de paquetes. La solución simula un sistema multiagente donde estos robots buscan, recogen y transportan paquetes de manera autónoma y colaborativa, gestionando eficientemente sus rutas y consumo energético. Se busca demostrar que los robots cumplan los flujos de trabajo requeridos, minimizando los tiempos de espera y evitando colisiones y cuellos de botella, todo visualizado en una simulación 3D que permite evaluar y ajustar el sistema.**

***Index Terms*—Multiagentes, Modelado Gráfico, Simulación, Unity, Sistemas Multiagente, LGV, Aprendizaje por Refuerzo.**

I. Introducción

SE propone realizar una simulación que represente el movimiento de un robot de transporte automatizado en un entorno industrial. Esta simulación permitirá al socioformador Electtric80 gestionar eficientemente las entregas en el entorno de trabajo, mejorando la precisión y optimización en las rutas de entrega [1].

I-A. Contexto y Problema

El problema particular radica en mejorar el movimiento de un robot en un ambiente de almacén. En la actualidad, se realizan estudios que investigan técnicas de optimización de rutas, como algoritmos A*, Q-learning y algoritmos para la planificación de acciones. La clave de estos sistemas radica en comprender restricciones espaciales (pasillos, zonas de carga y descarga) y temporales (velocidad del robot, periodos de carga), además de elementos como barreras estáticas y dinámicas.

I-B. Objetivos generales

El objetivo principal es crear una simulación que mejore las rutas de distribución dentro del almacén, mejorando la eficiencia y precisión de los robots en un entorno con múltiples restricciones. Objetivos concretos:

- **Objetivo 1:** Diseñar un modelo computacional realista que simule el entorno del almacén y el movimiento del robot, considerando las restricciones espaciales y temporales de dicho entorno.
- **Objetivo 2:** Implementar una simulación en Unity del sistema logístico automatizado con robots, optimizando el transporte de pallets, maximizando las entregas, y logrando una tasa óptima de operacion.
- **Objetivo 3:** Evaluar la eficacia de la simulación a través de contrastar los resultados obtenidos en la simulación (como tiempo de recorrido y distancia cubierta) con métricas estándar de la industria.
-

I-C. Restricciones

La simulación debe de considerar restricciones específicas, considerando dos ramas principales, espaciales y temporales

- **Espaciales:** Entre las restricciones espaciales tenemos las zonas de carga y descarga, ya que están limitadas a áreas específicas designadas, la capacidad de las estanterías, ya que estas solo pueden soportar tres pallets por nivel, además se deben de considerar los obstáculos dinámicos y estáticos, los robots deben ser capaces de detectar y evitar otros robots, además de las estructuras fijas como paredes, estanterías o zonas de carga.
- **Temporales:** Entre las restricciones temporales tenemos el tiempo de operaciones en la carga y descarga en las zonas de flujo gravitacional, el tiempo de carga y descarga de las baterías del robot, cada uno debe de detener su misión y buscar una estación de carga si su nivel de batería cae por debajo del 50 %, además que la duración máxima de la simulación es definida por el usuario.

Los abajo firmantes, {Alvaro Eduardo Lozano Medina, Luis Santiago Sauma Peñaloza,
Erick Segura Sánchez, Santiago De la Riva Juárez, Isaac Hernández Pérez}, declaramos que hemos cumplido a cabalidad con todos los requerimientos académicos y éticos exigidos por el Tecnológico de Monterrey. Afirmamos que nuestro trabajo en este proyecto ha sido realizado con respeto, honestidad y profesionalismo, en colaboración plena con el equipo, sin que haya existido ningún tipo de conflicto de interés o personal que afecte nuestra participación o la del equipo en conjunto. Este reporte ha sido firmado el día 29 de noviembre de 2024.

Alvaro Eduardo Lozano Medina

Luis Santiago Sauma Peñaloza

Erick Segura Sánchez

Santiago De La Riva Juárez

Isaac Hernández Pérez

I-D. Resumen de la solución propuesta

La solución se basa en un modelo multiagente que simula el comportamiento de un robot en un almacén, utilizando A* para la planificación de rutas. Unity se empleará para visualizar el movimiento y validar la eficacia de la solución.

II. Fundamentos

Se deben describir muy brevemente los conceptos fundamentales utilizados para la realización del proyecto. Es importante no olvidar citar los trabajos consultados.

II-A. Ecuación de Bellman

La ecuación de Bellman, dada por

$$Q(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a'), \qquad (1)$$

expresa el valor esperado de tomar una acción a en un estado s, seguida de la mejor política posible en los estados futuros. Aquí, $r(s,a)$ es la recompensa inmediata al realizar la acción a en el estado s, γ es el factor de descuento que pondera las recompensas futuras, $P(s'|s,a)$ es la probabilidad de transición del estado s al estado s' dado que se toma la acción a, y $\max_{a'} Q(s',a')$ es el valor máximo futuro esperado del mejor Q-valor posible en el estado s'.

Esta ecuación es fundamental en los métodos de optimización de políticas, como Q-learning, donde se busca maximizar la suma de recompensas futuras.

III. Descripción del Sistema Multiagente

El sistema multiagente desarrollado está compuesto por robots autónomos guiados por láser (LGVs) que operan de manera colaborativa para gestionar la logística y el transporte de paquetes en un entorno de almacén. Estos robots actúan como agentes individuales que poseen capacidades de percepción, decisión y acción, lo que les permite identificar paquetes, calcular rutas eficientes y evitar colisiones con otros agentes en tiempo real. Además de los robots, el sistema cuenta con estaciones de carga, que proporcionan energía a los LGVs, estanterías de almacenamiento y paquetes, que ofrecen ubicaciones específicas para los paquetes. Estos agentes, junto con los paquetes, forman una red de interacción que permite a los robots optimizar sus tareas de transporte y gestión de recursos en el almacén.

III-A. Modelo de los Agentes

El agente central de este sistema es el robot de transporte automatizado (LGV), cada robot está modelado como un agente autónomo con un conocimiento parcial del entorno, lo que incluye la ubicación de los paquetes (puntos de recolección y entrega), además de la posición de otros robots cercanos. Los planes de cada robot abarcan tareas como la recolección, transporte y entrega de paquetes, y se ajustan dinámicamente a la información percibida en tiempo real, lo que permite a los robots adaptarse a cambios en su entorno. Además, los robots cooperan entre sí compartiendo datos

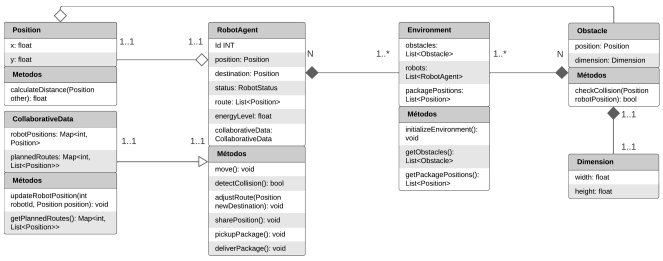

Figura 1: Diagrama de agente (robot)

de posición y rutas planeadas para evitar conflicto en áreas congestionadas, reduciendo la probabilidad de colisiones.

La medida de rendimiento de estos robots se evalúa en función a su eficiencia en el transporte de paquetes, minimizando los tiempos de espera y optimizando el consumo energético. Esta medida también considera la capacidad de los robots para evitar colisiones y mantener un flujo continuo en el almacén. Para lograr esto, los robots están equipados con sensores de proximidad que les permiten detectar obstáculos, así como sensores de ubicación que les facilita identificar puntos de recolección y entrega. Los actuadores incluyen a las ruedas omnidireccionales, que les permite desplazarse en cualquier dirección con precisión, y un mecanismo de manipulación para recoger y soltar paquetes.

Por otro lado las estaciones de carga, funcionan como agentes pasivos y reactivos en el sistema. Su único propósito es brindar energía a los robots cuando estos detecten que su nivel de batería es bajo. La ubicación de estas mismas en el almacén influirá en la planificación de rutas de los robots, ya que estos deben acudir a ellas en momentos críticos para mantener su nivel de operatividad.

Las estanterías de almacenamiento o racks, también son agentes pasivos y estáticos, diseñados para ofrecer ubicaciones específicas de almacenamiento de paquetes dentro del almacén. Estas estanterías limitan el número de paquetes que se puede almacenar en cada nivel, por lo cual debe ser considerado en las rutas de los robots.

Otro agente muy importante son los paquetes, que funcionan como agentes completamente pasivos, cuyo único rol es ser transportados dentro del almacén. Los paquetes no tienen capacidades de percepción ni acción; dependen completamente de los robots para su traslado.

III-B. Modelo del Entorno

El entorno del almacén está preconfigurado con obstáculos fijos y rutas definidas, proporcionando una estructura básica de navegación para los robots. Es un entorno parcialmente observable, dinámico y cooperativo, ya que los robots deben adaptarse constantemente a la presencia y movimientos de otros agentes, compartiendo información para evitar colisiones y minimizar cuellos de botella. La segmentación espacial en celdas de movimiento representa los pasillos, estanterías y zonas de carga, facilitando el cálculo de rutas eficientes. Además, el tiempo se maneja en intervalos

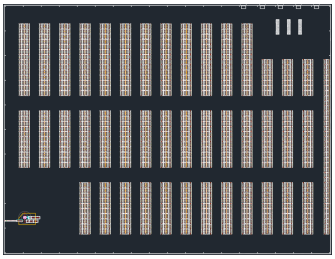

Figura 2: Graficación del entorno (.dwg)

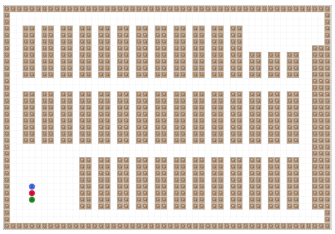

Figura 3: Implementación inicial del entorno en MESA

discretos, lo que permite una coordinación precisa entre los robots y facilita la simulación de interacciones en el entorno complejo del almacén

III-C. Modelo de Negociación

En el sistema multiagente del almacén, los robots LGV interactúan de manera cooperativa para evitar conflictos de ruta y minimizar los tiempos de espera en áreas congestionadas. La comunicación entre ellos se basa en mensajes simples sobre su posición actual, rutas planificadas y estado de misión. Cada robot comparte su ubicación y ruta con otros robots cercanos, permitiendo que estos ajusten sus movimientos en caso de una posible colisión. Además, los robots informan su estado de misión (en curso, completada o en espera) para que otros agentes puedan anticipar la disponibilidad de ciertas zonas.

III-D. Modelo de la Interacción

Cada robot transmite actualizaciones de su posición y estado a la interfaz gráfica en Unity, lo cual permite visualizar las rutas, los puntos y las posiciones relativas de los agentes en el plano. Esta visualización facilita el monitoreo en tiempo real de cada robot, mostrando los patrones de movimiento y ayudando a identificar áreas donde se podría optimizar aún más el flujo logístico. *Ver Figura 4*

IV. Descripción del Modelado Gráfico

IV-A. Escena a Modelar

Presente un borrador de la escena a modelar, seguido de la versión final en Unity. Compare las expectativas con el resultado real. No olvide incluir gráficos y ecuaciones que faciliten la interpretación del modelo propuesto.

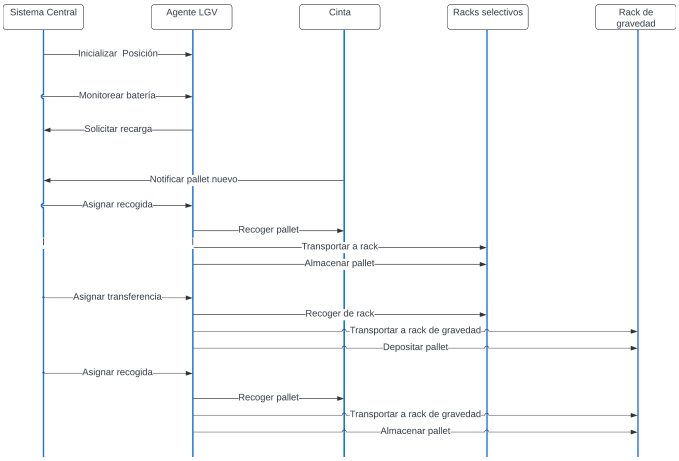

Figura 4: Diagrama de interacción

V. Algoritmo A*

Se muestra el Pseudocódigo 2 a título de ejemplo de cómo se incluye la descripción de un algoritmo utilizado para dar solución al reto.

Pseudocódigo 1 Algoritmo de Búsqueda A*

Require: Grafo $G = (V, E)$, nodo inicial s, nodo objetivo g, función heurística $h(n)$
Ensure: Camino más corto desde s hasta g
Inicializar la lista abierta $\mathcal{O} \leftarrow \{s\}$
Inicializar la lista cerrada $\mathcal{C} \leftarrow \emptyset$
Establecer $g(s) \leftarrow 0$, $f(s) \leftarrow h(s)$, y el padre de s como nulo
while $\mathcal{O} \neq \emptyset$ **do**
    Seleccionar $n \in \mathcal{O}$ como el nodo con el menor $f(n)$
    if $n = g$ **then**
        return Reconstruir el camino desde s hasta g
    Eliminar n de $\mathcal{O}$ y añadir n a $\mathcal{C}$
    for cada vecino m de n **do**
        if $m \in \mathcal{C}$ **then**
            Continuar con el siguiente vecino
        Calcular $g(m)$ tentativo $\leftarrow g(n) + \text{coste}(n, m)$
        if $m \notin \mathcal{O}$ o $g(m)$ es menor que el $g(m)$ anterior **then**
            Asignar el padre de m a n
            Establecer $g(m) \leftarrow g(n) + \text{coste}(n, m)$
            Establecer $f(m) \leftarrow g(m) + h(m)$
            if $m \notin \mathcal{O}$ **then**
                Añadir m a $\mathcal{O}$
return Fracaso, no se encontró ningún camino

procedure RECONSTRUIR CAMINO(g)
    Inicializar el camino como una lista vacía
    Establecer el nodo actual como g
    while el nodo actual tiene un padre **do**
        Insertar el nodo actual al inicio del camino
        Establecer el nodo actual como el padre del nodo actual
    Insertar s al inicio del camino
    return camino

V-A. Componentes Gráficos

- **Nombre del Componente 1**: Breve descripción y fuente. Se debe hacer referencia a su imagen correspondiente en la Figura **??**

V-B. *Prefabs*

- **Palet**: La funcionalidad de los palets es simular palets reales. Estos cuentan con una clase prefab que incorpora diversas funciones: una para determinar si el palet está siendo transportado (transportado), otra para identificar qué agente lo está transportando (transportado por) y, finalmente, una función para indicar si el palet ya ha sido entregado (delivered).

V-C. *Scripts*

Describa cada script y sus interacciones con otros elementos del proyecto. Incluya la fuente si se reutilizó código. Utilice el mismo formato que en el Pseudocódigo 2.

Tenemos un solo script a* en el que implementamos todas las funciones necesarias para nuestro proyecto:

- **CentralControlAgent**: El agente central se encarga de gestionar los agentes moviles, este tiene la responsabilidad de crear un grafo de navegación, planificar las rutas optimas para cada uno, y liberar las resevar para los robots que necesitan replantear su ruta, este utiliza networkx para construir y gestionar el grafo de navegación.
- **MovingAgent**: Representa a los robots móviles que recogen los pallets y los tranportan a su respectivo destino, se encargan del manejo de bateria en caso de que este bajo, detectan las colisiones solicitando replanificacion, lo que realiza es agente es solicitar las rutasy colabora con los agentes BaterryCharger.
- **MultiAgentModel**: El MultiAgentModel se encarga de inicializar el mapa, basado en descripción, crea y asignas los agentes MovingAgent, CentralControlAgent y PalletAgent, genera nuevos pallets, y realiza actualizaciones globales este se inspira en el framwork de mesa.
- **FromFescToMaze**: Se encarga de transforma la decripción en una lista de posiciones, este utiliza MultiAgentModel para realizar la configuración y las posiciones de la descripción se implementan a sus respectivos agentes.
- **BatteryCharger**: Se encarga de representar los puntos de baterias que hay en el mapa, estos solo pueden ser utilizados por un robot a la vez y aumenta el nivel bateria del robot.
- **PalletAgent**: Este representa a los pallets que van a ser tranportados por los robots y es gestionado por MovingAgents, también pueden ser generados por MultiAgentModel.
- **Server Mesa**: Muestra gráficamente los agentes, muestra el número de las colisiones, genera gráficos individuales para minotorear el nivel de bateria de los robots y actualiza los datos en tiempo real en DataCollector.

VI. Administración del Proyecto

- Vínculo al Product Backlog
- Vínculo al Repositorio de Github

Pseudocódigo 2 Algoritmo de Búsqueda A*

Require: Grafo $G = (V, E)$, nodo inicial s, nodo objetivo g, función heurística $h(n)$
Ensure: Camino más corto desde s hasta g

Inicializar la lista abierta $\mathcal{O} \leftarrow \{s\}$
Inicializar la lista cerrada $\mathcal{C} \leftarrow \emptyset$
Establecer $g(s) \leftarrow 0$, $f(s) \leftarrow h(s)$, y el padre de s como nulo
while $\mathcal{O} \neq \emptyset$ **do**
    Seleccionar $n \in \mathcal{O}$ como el nodo con el menor $f(n)$
    if $n = g$ **then**
        return Reconstruir el camino desde s hasta g
    Eliminar n de $\mathcal{O}$ y añadir n a $\mathcal{C}$
    for cada vecino m de n **do**
        if $m \in \mathcal{C}$ **then**
            Continuar con el siguiente vecino
        Calcular $g(m)$ tentativo $\leftarrow g(n) + \text{coste}(n, m)$
        if $m \notin \mathcal{O}$ o $g(m)$ es menor que el $g(m)$ anterior **then**
            Asignar el padre de m a n
            Establecer $g(m) \leftarrow g(n) + \text{coste}(n, m)$
            Establecer $f(m) \leftarrow g(m) + h(m)$
            if $m \notin \mathcal{O}$ **then**
                Añadir m a $\mathcal{O}$
return Fracaso, no se encontró ningún camino

procedure RECONSTRUIR CAMINO(g)
    Inicializar el camino como una lista vacía
    Establecer el nodo actual como g
    while el nodo actual tiene un padre **do**
        Insertar el nodo actual al inicio del camino
        Establecer el nodo actual como el padre del nodo actual
    Insertar s al inicio del camino
    return camino

VII. Resultados

Presente los resultados obtenidos en la simulación, comparando con los objetivos propuestos. Incluir gráficos o tablas si es necesario.

VIII. Conclusión

Resuma los principales hallazgos del proyecto y la efectividad de la solución propuesta. Comente posibles mejoras y limitaciones encontradas.

IX. Trabajo Futuro

Mencione posibles direcciones para continuar el desarrollo del proyecto en el futuro, basándose en las limitaciones observadas.

Referencias

[1] G. Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.