INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY,
CAMPUS GUADALAJARA

Tecnológico de Monterrey

Integrating Activity

By

Diego Rodríguez Romero A01741413

Diego Alejandro Calvario Aceves, A01642806

Milan Alejandro De Alba Gómez A01637667

Gonzalo Calderón Gil Leyva A01740008

Modeling of Multi-Agent Systems with Computer Graphics, Gpo 301

Prof. Ivan Axel Dounce Nava

November 17th 2024

Integrating Activity

This simulation is designed to emulate a warehouse environment where multiple autonomous robot agents collaborate to organize objects into designated stacks. The goal of each robot is to efficiently sort objects by detecting, picking up, and stacking them while avoiding collisions and maintaining an organized grid layout. Agents are equipped with different sensors and can perform different movements like turns and advancing.

Agent Specifications

Each robot agent is defined by several properties and behaviors, enabling it to navigate, interact with objects, and make decisions autonomously within the warehouse grid. The agents have some properties, some of them are the following:

- Position(x,y): The coordinates on the grid of the agent
- Direction: The agent orientation which could be NORTH, SOUTH, EAST, WEST. Represented as ontology individuals to standardize directions.
- ID: An unique identifier for each agent
- Is_Carrying_Object: To know if the agent is carrying an object or not.
- Target Stack: Coordinates of the stack where the agents wants to put its object

Each agent also has a set of behaviors that can be defined as:

- Movement: The agent can move forward and make turns in any direction.
- Object Detection: Can use their sensor to detect objects, other agents or walls adjacent to them.
- Object Pickup: If empty, robots can pick up an object
- Collision Avoidance: Agents will adjust their direction if they sense that they will collide, this turns could be random.

-Agent Architecture

The Robot Agent operates with an hybrid architecture that combines reactive and deliberative components. This allows the agents to balance real-time reactions to environmental changes (such as avoiding obstacles and interacting with objects) with goal-oriented, deliberative actions (such as planning to organize the warehouse).

-Reactive Layer

It is responsible for the robot's real-time interactions with the environment. These responses are quick and do not involve complex planning or long-term decision-making. Instead, this layer focuses on sensing and responding immediately to the current state of the environment.

Between their primary functions are:

- Obstacle Avoidance**:** The robot detects adjacent obstacles or other robots and makes adjustments to avoid collisions.
- Immediate Object Detection and Pickup: When an object is detected on an adjacent cell, the robot quickly responds by attempting to pick it up if it's not already carrying one.
- Basic Movement: The robot uses simple movement rules, such as turning left or right if blocked or making random turns to continue navigating the grid.

-Deliberative Layer

The deliberative layer focuses on achieving the agent's longer-term goals, such as organizing the objects into stacks. For example:

- Goal Setting and Monitoring: The robot evaluates at each step whether the warehouse is organized. If carrying an object, it determines an appropriate destination stack based on stack capacity and position.
- Object Placement Decision-Making: For robots carrying an object, the deliberative layer determines where to place it. The robot assesses nearby stack capacity and selects an optimal placement if the stack is full.

In this hybrid architecture, both layers operate in parallel, with the reactive layer taking precedence in immediate situations that require fast responses, for example avoiding collisions, while the deliberative layer guides actions that require planning and alignment with the overall organization goal

Environment Specifications

The warehouse environment is modeled as a grid where each cell represents a location that can contain either an object, an empty space, or a stack. Here are the primary environmental features:

- Grid Dimensions: Defines the warehouse size.
- Obstacle placement: Initialize the objects at random positions at the beginning of the simulation.
- Stacks: Cells where up to 5 objects can be placed on top of each other

The environment also has an ontology integration that formalizes robot directions, stack states, and robot-object interactions. This enhances the simulation's flexibility and allows agents to dynamically interpret and interact with environmental conditions.

Alternative Solutions

For this simulation there exist a wide range of approaches that we could have used. We opted for a very classical approach for finding the objects, where we let the agents move freely through the grid and change direction only if a collision happened, if they sensed an object adjacent to them, or if they had been in the same path for too long. Also we added a 10% chance of making random turns which led to some interesting results. There were times that such a random turn ended up moving the robot away from an object or stack in their path, but most of the time it ended up putting the robot in the way to pick up an object. We think that implementing a path finding algorithm could be a better approach to solve these problems, since this could, very surely, have reduced the number of steps needed to organize the warehouse. Now, for the stack finding a pathfinding algorithm could have been useful as well, however, we defined that all the stacks must be created next to a wall or next to a previously created stack. Knowing that, only traveling the perimeter in any direction, clockwise or counterclockwise, could have produced equal or better results than those obtained.

Another improvement can be adding some sort of memory to the agent, this way we could store the information of the cells we have visited and its content, thus we can use the cells that the agent knows are empty only to get to another point and not be there exploring, and if the agent know the position of an stack it could go there and try to drop an object.

Computer Vision

Complementing the multi-agent system, a computer vision system was integrated to enable robots to visually identify objects and stacks in the environment.

1. **Camera Simulation**:
    - Each robot is equipped with a simulated camera that captures a field of view in front of it.
    - The FOV is converted into a dummy RGB image, simulating what a physical camera would capture.
2. **Vision Model Integration**:
    - The YOLO vision model is used to process the FOV image and identify objects or stacks in real-time.
    - The model outputs class labels (like "object" or "stack") and confidence scores for detected items.
3. **Signal for Detections**:
    - Robots provide feedback when detection:
        - Console Output: For example, "Robot 0 detected an object with confidence 0.85."
4. **Decision-Making**:
    - Robots use detection results to:
        - Pick up objects detected in their FOV.

- Navigate and interact with stacks based on visual input.

Impact of Computer Vision implementation

- **Improved Interaction**: Robots can visually identify objects and make informed decisions based on their environment.
- **Hybrid Logic**: Combines reactive actions with goal-setting for improved efficiency and adaptability.

Title		
Robot	Warehouse	Cell
State Description		
- Position: (x, y) - ID: Unique robot identifier - Carrying Object: True/False - Direction: One of NORTH, SOUTH, EAST, WEST - Movement History: List of recent positions	- Dimensions: (Width x Height) - Grid: 2D matrix of Cells - Robots: List of robot entities - Stacks: Tracks positions and heights of object stacks - Step Counter: Number of simulation steps	- Types: Empty, Stack, Wall, Object - Stack Height (if applicable): Maximum and current number of objects in a stack
Actions		
- sense_environment: Detects cell types in proximity - move_forward: Advances based on the current direction - pick_up_object: Picks up an object if available - drop_object: Drops the carried object - detect_objects: Identifies objects using a vision model	- add_robot: Places a robot into the warehouse - step: Simulates a single tick of the system - is_organized: Checks if all objects are correctly stacked	- None (passive elements in the warehouse)
Methods		
Methods - choose_direction: Decides next direction based on surroundings - update_position_history: Tracks movement history - act_on_detections: Reacts to objects detected in the field of view	- _place_objects: Distributes objects randomly in the warehouse - _initialize_robots: Assigns initial positions to robots - visualize: Displays the current warehouse state	- None (used conceptually or as part of the warehouse grid)

Success Metrics

The metric we defined for our simulation was focused on the efficiency of it. The average steps taken to place an object in a stack is our metric, we calculate it by adding the total number steps each robot performed and dividing it by the number of objects that were initialized in the simulation. Fewer steps per object placement indicate higher efficiency, reflecting minimal movement and effective pathfinding.