ASSIGNMENT – 3

ADVANCED ARTIFICIAL INTELLIGENCE (CSL7580)

Expert System for Autonomous Embodied Robots in A Warehouse Environment

TEAM:

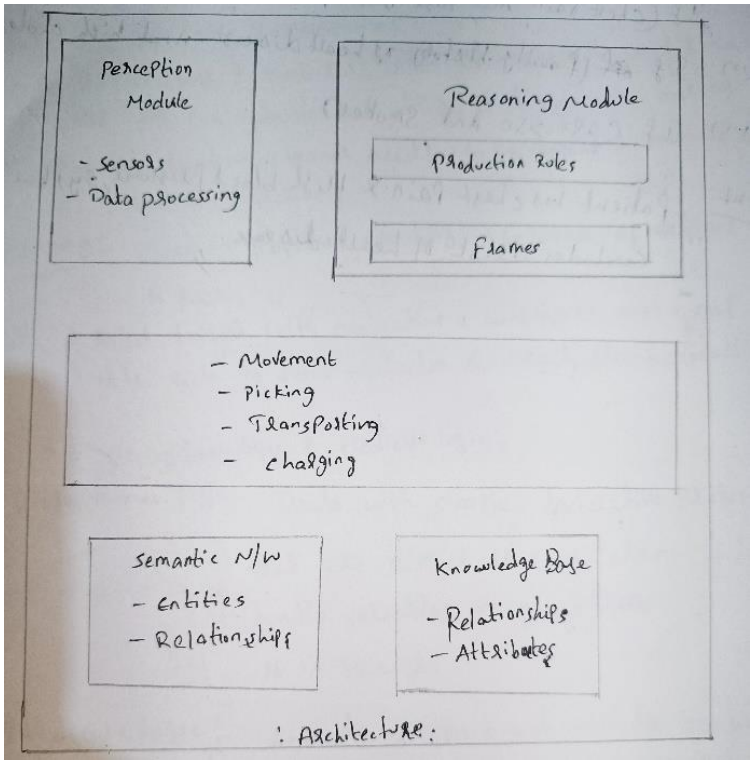
KRISHNA KUMARI RAVURI	M22AI567
RAKESH REDDY KADAPALA	M22AI608
CHAITRASHREE C	M22AI539
BHARATHI SONTEM	M22AI536
J.V.S KALYAN AASHISH	M22AI561

Q1. Design an expert system for autonomous embodied robots in a warehouse environment that utilizes traditional knowledge representation techniques, including semantic networks, frames, and production rules.

Knowledge representation & Planning

Ans. For an expert system architecture of a warehouse robot, there are various interconnected components that support efficient navigation, decision making and task execution. These include environment representation, robot state management, decision-making processes, action planning and integration mechanisms.

Environment Representation (Semantic Network):
For a robot to understand its surroundings, it requires environment representation as one of the most important aspects. A semantic network is used in modeling the warehouse environment where the robot can easily navigate and interact with different entities.



Nodes

- The locations: Are different parts within the warehouse such as corridors, shelves, charging stations among others. These places could be seen like a patchwork quilt of a store house.
- The items: These represent individual products carried by the warehouse linked to specific places there in.
- Obstacles: They are physical barriers or things that prevent or hinder the robot from moving during navigation.
- Charging Station: This represents an area where re-charging of batteries takes place.

Relationships: There exist several relationships among nodes in this semantic network through which nodes help define how certain entities work together:

Connected To: Indicates that two locations are next to each other or accessible via one another.
Contains: It means that some items are in specified areas such as aisles.
Is An Obstacle: Identifies which nodes are obstacles, guiding the robot to avoid them.

Robot State (Frame): The robot is represented by frames, that define the current status and characteristics of the robot. This representation enables the updates during the interaction of the robot with its surroundings.

Slots

- **Current Location:** Infers the location of the robot in the warehouse at any given time.
- **Battery Life:** Shows how much battery power is left, this is essential in estimating movements and other activites.
- **Carrying Status:** Displayed what item, if any, that was with the robot.

Dynamic Updates
The position of the robot changes its location within the warehouse the values on the slot changes as well. For instance, if the robot is carrying an item, the carrying status changes instantly and the latter is recorded.

Decision-Making (Production Rules): The goal selection and the process leading to it is very much based on production rules, which are indeed If-Then statements with a variety of context conditions and details of the robot's environment in the If part and the actual action in the then part.

Example Rule:
Rule: IF (location = cell X) AND (cell Y linked to cell X) AND (battery level more than or equal to cost of moving to Y) THEN go to cell Y.

- This rule shows that if the robot is in cell X currently and most energy is still remaining to make the transition, it must go to the connected cell Y.

Triggering Conditions: By the state of a robot and information contained within a semantic network the rules are initiated. This helps the robot to plan and decide as to what it has to do next or where it has to go.

Action Planning (Frames): Action planning implies the identification of some courses that the robot can take given its status and the surroundings. The same is represented within frames.

Action Frames

- **Potential Actions:** Include the Body movements like up, down, left, right and the operation movements like pick up and drop.
- **Preconditions:** Subprocesses located in action frames define conditions that should be met for the action to be performed (ex: enough battery charge).
- **Postconditions:** Slots define what is expected to happen next to the action (e. g. a new location, updated battery level).

Integration: Flexibility is therefore important to enable the robot to work in an environment where other components are already present. This is the process of coordinating each of the components of the expert system.

Sensor Input: The robot uses sensors to gather information about its surroundings. This data is processed and used to update the semantic network, ensuring that the robot has an accurate understanding of its environment.

Status Updates: The robot's status frame is continuously updated based on the actions it takes. For example, if the robot moves to a new location, its current location slot is updated accordingly.

Action Selection: The decision-making module assesses the current state (as represented in the semantic network) against the production rules to select the most appropriate action based on environmental conditions.

Preconditions Verification: Before executing an action, the robot checks whether the preconditions specified in the action frames are met. This ensures that the robot only attempts actions that are feasible given its current state.

Handling Ambiguity and Incomplete Information:

Default Values: In situations where the robot encounters unknown cells in the warehouse, default or 'empty' values are assigned in the semantic network. This allows the robot to proceed with its tasks while still exploring and updating its knowledge.

Uncertainty Factors: To manage uncertainty, the system associates certainty values with the links in the semantic network. This reflects the confidence levels regarding the correctness of the information, allowing the robot to make more informed decisions.

Exception Handling Rules: When the robot encounters an obstacle or situation not represented in the semantic network, exception handling rules are triggered. These rules initiate new planning processes, allowing the robot to adapt to unexpected circumstances.

Example Scenario: When the robot discovers an 'empty' cell in the warehouse, it updates the semantic network to reflect this new information. For instance, if it finds a previously unknown item in that cell, the network is updated to include the item's details, and the robot's carrying status is adjusted accordingly.

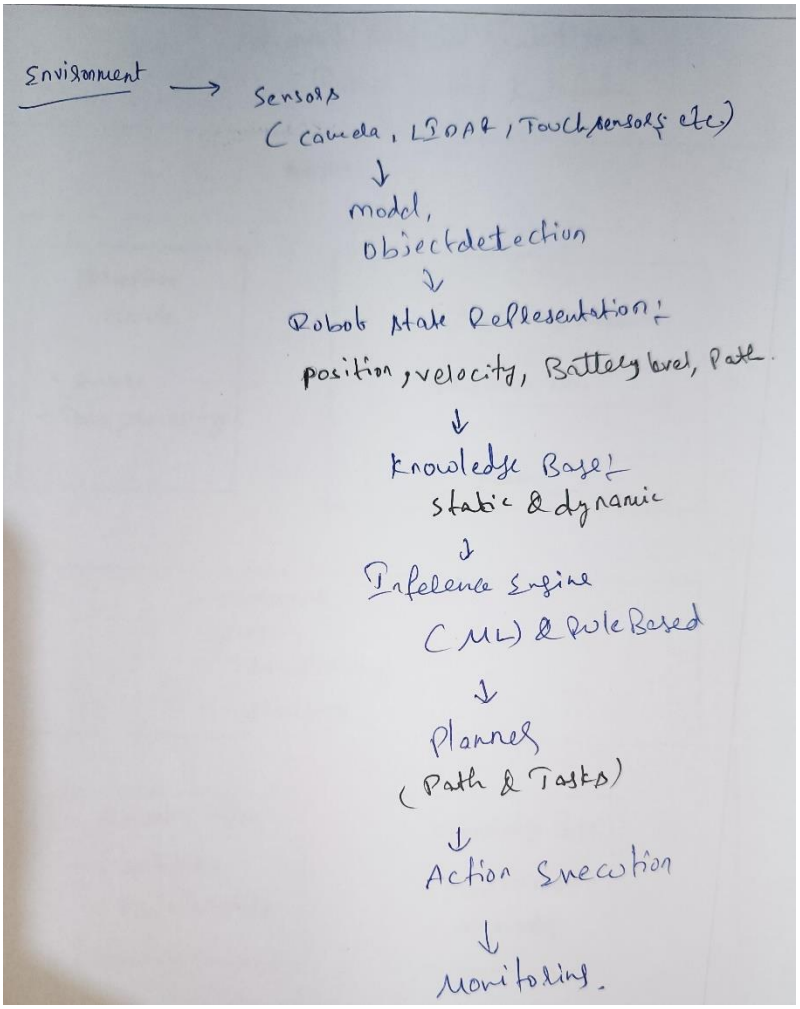
Q. Draw the System Architecture: Clearly indicate how each knowledge representation method is integrated into the expert system for the robot. Include components such as environment representation, robot state, decision-making, and action planning.

Ans.

Environment Representation

Sensors:

- **Cameras:** Obtain information about the surroundings in the form of images/frames/graphics which are used for identification, localization and monitoring of objects and possible barriers.
- **LIDAR** (Light Detection and Ranging): The distance is measured based on the Laser light emitted reflecting back to the sensor thus giving detailed three-dimensional maps of the environment surrounding the robot.
- **Ultrasonic Sensors:** Utilize sound waves as the primary way to perceive the environment and its distance, which is helpful for obstacle avoidance at short distances.
- **Touch Sensors:** Avoid position and distance, competent for indications of touching of objects for feeling and moving around.
- **GPS** (Global Positioning System): Gives global location information, which is crucial in the outdoors.



Mapping: A type of visualization and modeling methods, SLAM methods are used to continuously build a map of the environment as a robot navigates in it.

Object Detection and Classification: Maps objects in the environment such as people, other robots, etc and furniture and other similar items by employing the concept of machine learning.

Robot State Representation

- Position: The current position and direction of the robot in the environment in the form which is acquired from odometry or GPS or SLAM.
- Velocity: Robot orientation and velocity that refers to the speed and direction in which the robot can move and which can be regulated depending on the task.
- Battery Level: Battery charge remaining, to avoid the robot running out of charge and becoming useless for unknown how long to charge again.
- Internal Health: Condition of internal peripherals such as the sensors and the actuators, to make sure that they are in order.

Knowledge Base representation:

Static Knowledge:
Rules: Conditional statements that were preprogrammed to determine the steps to be taken by the robot given circumstances. For example, possible rule is “If there is an object within 1 meter, then interrupt the maneuver and turn around.
Objectives: The objectives that the robot has to accomplish; for instance; “get the package to the specified destination” or “monitor the area”.

Dynamic Knowledge:
Learning Algorithms: Some of the machine learning models that enable the robot to update its stock of knowledge based on the actions carried and their results. For instance, an aspect of machine learning such as, reinforcement learning means that over time the robot can learn which routes to take.

Decision-Making:

Rule-Based System: Uses predefined rules on the current state and environment model to take decisions. For instance, it may utilize rules to decide how to avoid obstacles or when to charge the battery.

Machine Learning Models: These models anticipate the most appropriate action based on historical data. For example, a neural network might be taught to identify optimum navigation paths in different settings.

Action Planning:

- Path Planning Algorithms: Such algorithms determine the best way from where the robot is to its destination, taking into account barriers and also considering abilities and restrictions of the robot’s performance.
- Task Planning: Creates an order of tasks that should be carried out so as certain high-level goals can be achieved, such as “Pick up the item, navigate to the delivery point, place item.”

Action Execution:

Actuators: The motors, servos, and other devices responsible for carrying out motions programmed by our planner include walking around, manipulating objects, dealing with environment.

Workflow:

Environment Perception: Processing the collected information about surrounding by sensors that continuously monitor it leads to updating of environmental model.

- **State Update:** The robot's state variables are updated in real-time based on sensor data and internal diagnostics.
- **Knowledge Application:** The inference engine uses the knowledge base to interpret the current state and environmental model, making decisions about what actions to take.
- **Planning:** The planner takes the decisions from the inference engine and creates a detailed action plan, including path planning and task sequencing.
- **Action Execution:** The planned actions are sent to the robot's actuators, which perform the necessary movements and operations to achieve the objectives.
- **Feedback Loop:** The results of the actions are fed back into the system through the sensors, updating the environmental model and state representation, which influences future decisions

---

Q. Handling Challenges of Ambiguous or Incomplete Information: Explain how your system handles the challenges of ambiguous or incomplete information within the warehouse environment, such as unknown obstacles or dynamically changing item locations. Provide examples to illustrate your approach.

Ans. Handling ambiguous or incomplete information in a warehouse environment involves several strategies and technologies to ensure smooth operations despite uncertainties. Items in these large warehouses are often times shifted about, there are shelves that need to be restocked and other maintenance type work that results in dynamic obstacles. An AMR takes goods from storage zones and delivers them to the packing islands. Based on the WMS system, the initial position of the items is given to the AMR, but due to changes in position frequently, the AMR meets a unfamiliar obstacle or learn that the items are not where they are supposed to be many times.

Real Time Data Capture along with Analytics: The AMR is fitted with LIDAR technology together with ultrasonic sensors and vision systems. These sensors are mounted on the AMR and these scan the environment with the aim of identifying any obstacles and at the same time updating the map that the AMR has on board.

- **Example:** Suppose the AMR is moving towards a storage location; it then identifies an object in the aisle that it was not expecting to see such as a pallet. Information is updated and the AMR recognizes the size and position of the obstacle.

Dynamic Path Planning Algorithms: To be specific, the AMR employs efficient and effective path planning where the movements' trajectory is adapted depending on the readings from the robot's perception system.

- **Example:** When identifying the pallet, the AMR's algorithm redesigns the kind of a path free from the detected obstacle. In the process it quickly identifies another route that it has to take in order to get to the target storage point that it has to reach without having to stop for human intervention.

Machine Learning and AI: The AMR follows a machine learning approach that is trained on the history of the layout of the warehouse and the usual types of interference to allow it to solve problems with incomplete information.

- **Example:** In the present case, the AMR's AI system will anticipate that certain aisle may have more obstacles in the course of the day because of restocking operations. Thus, it proactively adapts its route planning with the view to not include these hot spots during these periods which will minimize the probability of dealing with such impediments.

Human Direction Systems: Whenever the AMR is not able to decide on an ambiguous situation, it can seek guidance from a human operator.

- **Example**: If the AMR sees an obstacle that it cannot navigate around or classify, it immediately sends a warning to the human operator. The operator remotely views the AMR's camera feed and evaluates the situation then provide manual navigation instructions or physically remove the barrier.

Feedback Loops and Continuous Improvement: All encounters with obstacles and instances where robots do not follow their path are recorded by this robotic system in its computers for further analysis leading to continuous improvement.

For instance, After successfully going round this box, there will be an entry made of what happened at that time, where this was as well as how it realigned. With time this will be used for improving future path planning algorithms thus reducing similar disturbances.

Example: AMR (Autonomous Mobile Robots) in an E-Commerce Warehouse:
AMRs are very important in an e-commerce warehouse for moving goods from stores to packing stations even with constant changing of layout due to restocking and maintenance. These AMRs have LIDAR, ultrasonic sensors and cameras, which provide the real-time data necessary for identifying and avoiding objects like pallets. Employing dynamic path planning A* algorithm, these AMRs continually take sensor data into account when faced with unexpected obstacles so as to come up with new routes that are shorter than the previous ones. Models created using machine learning and trained on historical warehouse data can tell us where high-traffic zones may be during peak hours and thus redirect around them accordingly. In instances where a particular AMR faces insurmountable barriers, human operators get alerted who then remotely examine what is happening before giving directions or removing it. Improvement of algorithms through detailed logging of interventions and encounters enhances feedback loops in sensor integration. Such multi-pronged approach allows effective operation of AMRs within volatile contexts thereby reducing downtimes while improving productiveness as well as ensuring smooth running under uncertain situations involving incomplete or imprecise information.

By employing above strategies, the e-commerce warehouse ensures that its AMRs operate efficiently and adapt to dynamic and unpredictable environments. The combination of real-time data collection, dynamic path planning, machine learning, human-in-the-loop systems, and feedback loops minimizes downtime, enhances productivity, and ensures smooth operations despite the challenges posed by ambiguous or incomplete information. This approach not only improves the efficiency of AMRs but also enhances the overall productivity and reliability of the warehouse operations.

Q1-B: Consider the same autonomous embodied robot in the large warehouse designed to fetch items for shipping. The warehouse is organized into a grid where each cell can either be empty, contain an item, or be an obstacle. The robot can move up, down, left, or right and can pick up items when it is in the same cell as an item. The robot has a battery limit, which only allows it to make a certain number of moves before needing to return to a charging station.

Design a planning algorithm that allows the robot to optimize its path to collect a maximum number of items and return to the charging station before the battery runs out. Discuss the following aspects:

The representation of the environment and the robot's state.

The planning strategy you would use (e.g., heuristic search, greedy algorithms, A* algorithm).

How the robot can handle dynamic changes in the environment, such as newly placed obstacles or items.

Evaluate the efficiency of your proposed solution in terms of computational complexity and the quality of the path generated.

Q. The planning strategy you would use (e.g., heuristic search, greedy algorithms, A* algorithm)

Ans. **Environment Representation and the robot's state.**

Grid Layout: The warehouse is organized into a grid of square cells, with each cell having a specified width (GRID_SIZE).

- This grid simplifies the robot's movement plans by discretizing the space into manageable units.
- The grid is created using a function that generates coordinates for each cell based on the screen size and the grid size.

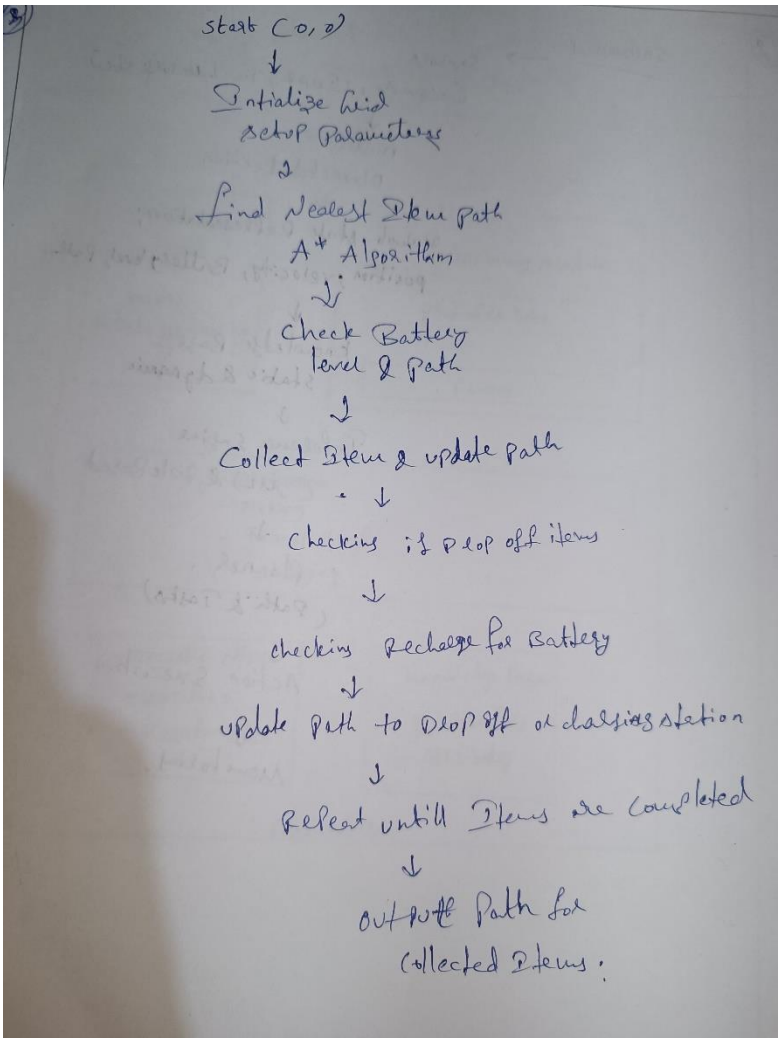
Obstacles:

- **Fixed Obstacles:** These are predetermined barriers placed at specific regions within the grid.
  - **Example:** Three fixed obstacles named Obstacle_1, Obstacle_2, and Obstacle_3.
- **Random Obstacles:** These obstacles are generated randomly within the grid, providing a dynamic environment and they are created by a function that ensures random placement without overlapping the fixed obstacles.

Special Locations:

- **Charging Station:** The location where the robot recharges its battery.
- **Storage Room:** The area where items are stored and picked up by the robot.

- Patient Room: The destination where the robot delivers items.



Pathfinding with A* Algorithm:

Heuristic Function:

Manhattan Distance: This heuristic is used to estimate the cost from the current position to the goal, suitable for grid environments with horizontal and vertical movements.

A* Algorithm:

- **Open Set:** A priority queue that stores nodes to be evaluated, prioritized by their estimated total cost (f_score).
- **Closed Set:** A collection of nodes that have already been evaluated to prevent reprocessing.
- **Path Reconstruction:** Once the goal is reached, the path is reconstructed by backtracking from the goal node using a dictionary that tracks node origins.

Algorithm Steps:

- Initialize the open set with the start node and calculate its f_score.
- While the open set is not empty:
    - Pop the node with the lowest f-score.
    - If this node is the goal, reconstruct and return the path.
    - For each neighboring node:
    - Calculate the tentative cost.
    - Update scores if the new path is better.
    - Add the neighbor to the open set if it is a viable path.
- If no path is found, return an empty list.

Robot State Management

Manages the robot's actions and task transitions:

- **Delivery_request:** The robot receives a request and moves towards the storage room.
- **Navigate_to_storage:** The robot navigates to the storage room to pick up items.
- **Pick_up_medication:** The robot collects items from the storage room.
- **Navigate_to_patient_room:** The robot moves towards the patient room to deliver items.
- **Deliver_medication:** The robot delivers items to the patient and returns to the charging dock.
- **Update_obstacles:** The robot updates its knowledge of obstacles to adapt to changes in the environment.

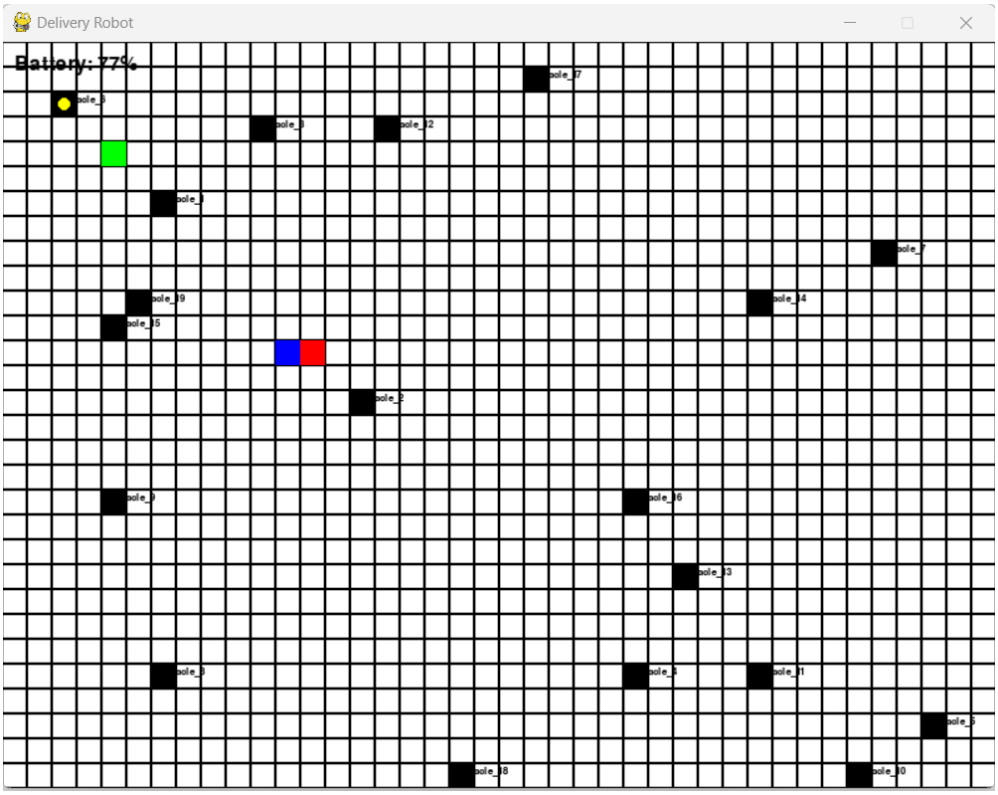
Battery Management:

Battery Monitoring:

- The robot monitors its battery level and decides when to recharge:
- If the battery level drops below a certain threshold (e.g., 30%) and the robot is not carrying items, it returns to the charging station.
- The robot recharges fully when at the charging station.

Obstacle Handling:

- The A* algorithm accounts for obstacles to minimize movements and find feasible paths.
- Obstacles are periodically updated (e.g., during the UPDATE_OBSTACLES state) to simulate changes in the environment and test the robot’s adaptability.




```
#  grid: 0 = empty, -1 = obstacle, >0 = item , 1,2,3 are items
grid = [
    [ 0,  0,  -1,  2,  0],
    [ 0,  1,  0,  0, -1],
    [ 0,  0,  0,  2,  0],
    [ 0,  0,  3,  0,  0],
    [ 0, -1,  0,  2,  0]
]
```

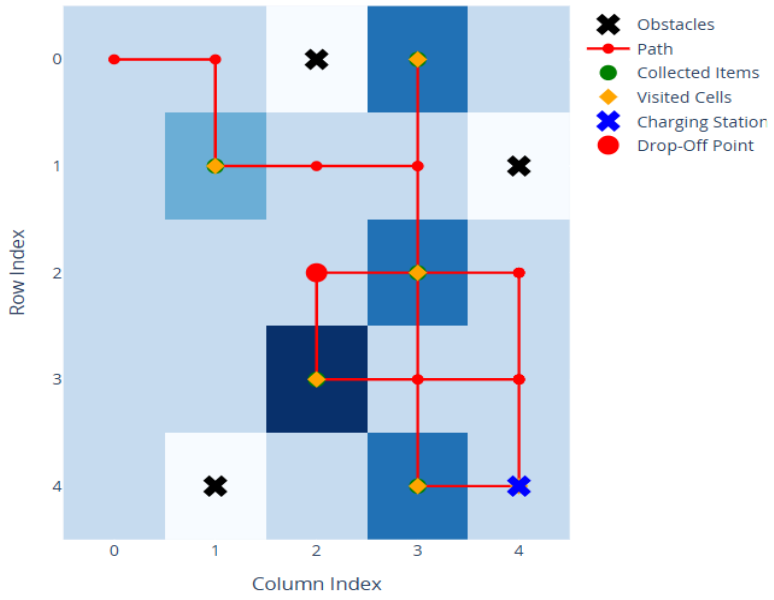


Fig: Planning algorithm of Robot in Warehouse Environment

```
Collected Items: ['Item Type 1', 'Item Type 2', 'Item Type 2', 'Item Type 3', 'Item Type 2']
Total Time Taken: 26.58 minutes
Total Charging Events: 3
Path to collect item at (1, 1): [(0, 0), (0, 1), (1, 1)]
Path to collect item at (4, 3): [(4, 4), (4, 3)]
Path to collect item at (2, 3): [(2, 2), (2, 3)]
Path to collect item at (3, 2): [(4, 4), (3, 4), (3, 3), (3, 2)]
Path to collect item at (0, 3): [(4, 4), (3, 4), (2, 4), (2, 3), (1, 3), (0, 3)]
Time taken to collect item at (1, 1): 4.92 minutes
Time taken to collect item at (4, 3): 3.90 minutes
Time taken to collect item at (2, 3): 5.53 minutes
Time taken to collect item at (3, 2): 7.57 minutes
Time taken to collect item at (0, 3): 1.67 minutes
Recharge 1: 1.00 minutes
Recharge 2: 1.00 minutes
Recharge 3: 1.00 minutes
```

Fig: Results of Robot collecting Items in Warehouse Environment

We can observe from above figures, how robot is collecting Items while avoiding the obstacles and delivering them at dropping point.

Q. How the robot can handle dynamic changes in the environment, such as newly placed obstacles or items.

To handle dynamic changes in the environment, such as newly placed obstacles or items, the robot can follow a structured approach that involves continuous monitoring, updating its internal map, and recalculating paths as needed.

Dynamic Environment Handling Strategy

Continuous Monitoring: The robot should continuously monitor its environment using sensors or communication with a central system that provides updates about new obstacles or items.

- **Sensor Integration:** Equip the robot with sensors (e.g., LIDAR, cameras) to detect obstacles and items in real-time.
- **Central System Updates:** Implement a communication protocol with a central control system that can provide updates about changes in the environment.

Internal Map Updates: The robot maintains an internal map of the environment, which includes the grid layout, fixed obstacles, and known locations of items. This map needs to be updated dynamically as new information is received.

- **Map Data Structure:** Use a data structure (e.g., a 2D array or graph) to represent the grid and store information about obstacles and items.
- **Update Function:** Implement a function to update the internal map with new obstacles or items based on sensor input or central system updates.

Path Recalculation: Whenever a change in the environment is detected, the robot should recalculate its path to its current goal using the updated map.

- **Pathfinding Trigger:** Set a trigger to initiate path recalculation whenever an obstacle or item is detected or removed.
- **A* Algorithm Recalculation:** Used the A* algorithm to find the shortest path to the goal, taking into account the updated map. This involves recalculating the heuristic values and re-evaluating the open set.

State Machine Adjustments: Incorporate states within the robot's finite state machine to handle dynamic changes and ensure smooth transitions between tasks.

- UPDATE_OBSTACLES State: Add a state to the state machine that deals specifically with updating the map and recalculating paths.
- Error Handling: Include error handling states or mechanisms to manage scenarios where the robot cannot find a feasible path due to dynamic changes.

Reactive Planning: It involves dynamically adjusting the robot's path in response to changes in its environment. When a new obstacle is detected, the robot immediately stops and evaluates its current path. Using local sensors, the robot recalculates a new route around the obstacle, often utilizing simpler algorithms like potential fields or local search methods. This approach focuses on short-term adjustments and is effective in environments where obstacles appear suddenly and frequently. The robot's onboard processing unit continuously monitors sensor inputs to trigger path recalculations and adjust movements in real-time.

Hybrid Planning: Hybrid planning integrates both global and local pathfinding strategies to handle dynamic environments. Initially, a global pathfinding algorithm like A* is used to determine the optimal route from start to goal. As the robot moves, a local obstacle avoidance algorithm, such as the Dynamic Window Approach (DWA) or Rapidly-exploring Random Trees (RRT), is employed to manage immediate obstacles. This approach allows the robot to benefit from a precomputed optimal path while adapting to unexpected changes in real-time.

Probabilistic Roadmaps (PRM): Probabilistic Roadmaps involve creating a network of feasible paths through random sampling. Initially, the environment is sampled to generate a roadmap consisting of nodes and edges that represent possible paths. When the environment changes, the robot uses this precomputed roadmap to quickly find new paths by adjusting connections in the graph. This method is particularly effective for large and complex environments where precomputing a roadmap can significantly reduce pathfinding time.

Q. Evaluate the efficiency of your proposed solution in terms of computational complexity and the quality of the path generated.

Ans.

Efficiency Evaluation of the Proposed Solution

A* Pathfinding Algorithm Complexity:

Time Complexity: The A* algorithm's time complexity is $O(b^d)$ where 'b' represents the branching factor (number of possible moves from each state), and 'd' is the depth of the search tree. In this application, the branching factor is 4 (corresponding to movements: up, down, left, right), and the depth depends on the grid's layout and obstacles.

Utilizing a heuristic function, such as the Manhattan distance, optimizes the algorithm by reducing the effective branching factor, enhancing its efficiency compared to uninformed search methods.

Space Complexity: The space complexity of A* is also $O(b^d)$ due to the storage requirements for the open and closed lists. For large grids or grids with many obstacles, this complexity can become substantial.

Pathfinding Operations Count: The robot's task involves several pathfinding operations for each item collection and recharge. Each item requires a pathfinding operation from the current location to the item and possibly to the drop-off or charging station. The total number of these operations affects the algorithm's overall efficiency and depends on the grid size and number of items.

Overall Time Complexity: The overall time complexity of the solution can be approximated as $O(n * b^d)$ where 'n' denotes the number of items or recharge events and (b^d) is the complexity of each pathfinding operation.

Path Efficiency:

- **Path Length:** The lengths of the paths taken to collect each item and return to the charging station vary. The paths range from as short as 2 cells to longer routes of up to 6 cells. The total path length provides an insight into how well the robot navigates while minimizing travel distance.
- **Path Quality:** The generated paths are effective, with the robot managing to avoid obstacles and return to the charging station as needed. This demonstrates a balance between item collection and battery management.

Charging Events:

- **Total Charging Events:** During the simulation, the robot underwent 3 charging sessions. This number suggests that the battery capacity was effectively utilized, with the robot making necessary recharges. Frequent recharges might imply that the battery capacity was somewhat limited, but it ensured that the robot did not deplete its battery.
- **Time Efficiency:** The total duration for the robot's task was 26.58 minutes. This encompasses the time for item collection and recharging. This measure reflects the overall efficiency of the robot's operations within the given time frame.
- **Item Collection Time:** The time taken to collect each item varied, which is indicative of the distance and difficulty in accessing each item. Efficient collection times imply that pathfinding was generally effective, though some paths may be longer due to the grid's constraints.

Recharge Duration:

The robot spent a total of 3 minutes on recharges. This duration is a small fraction of the total operational time, suggesting that recharge intervals were well-managed and did not significantly impact overall performance.

Results:

Collected Items: ['Item Type 1', 'Item Type 2', 'Item Type 2', 'Item Type 3', 'Item Type 2']
Total Time Taken: 26.58 minutes
Total Charging Events: 3
Path to collect item at (1, 1): [(0, 0), (0, 1), (1, 1)]
Path to collect item at (4, 3): [(4, 4), (4, 3)]
Path to collect item at (2, 3): [(2, 2), (2, 3)]
Path to collect item at (3, 2): [(4, 4), (3, 4), (3, 3), (3, 2)]
Path to collect item at (0, 3): [(4, 4), (3, 4), (2, 4), (2, 3), (1, 3), (0, 3)]
Time taken to collect item at (1, 1): 4.92 minutes
Time taken to collect item at (4, 3): 3.90 minutes
Time taken to collect item at (2, 3): 5.53 minutes
Time taken to collect item at (3, 2): 7.57 minutes
Time taken to collect item at (0, 3): 1.67 minutes
Recharge 1: 1.00 minutes
Recharge 2: 1.00 minutes
Recharge 3: 1.00 minutes

The Robot collected 5 Items in 27 Minutes accordingly mentioned by directions and charging conditions. For item 2 in (0,3) position robot took less time 1.67 Minutes.

References:

1. Keith, Russell & La, Hung. (2024). Review of Autonomous Mobile Robots for the Warehouse Environment.

2.Warehouse Automation: Types, Benefits, & More

https://www.exotec.com/insights/warehouse-automation-types-benefits-and-more/

3.The Road to Embodied AI

https://wayve.ai/thinking/road-to-embodied-ai/

4.Retrospectives on the Embodied AI Workshop

5.Autonomous mobile robots: warehouse applications and uses

https://www.mecalux.com/blog/autonomous-mobile-robots

6.5 ways autonomous mobile robots are transforming warehouses

https://6river.com/how-autonomous-mobile-robots-are-transforming-warehouses/

7.Grover, Abhay & Ashraf, Muhammad Hasan. (2023). Leveraging Autonomous Mobile Robots for Industry 4.0 Warehouses: A Multiple Case Study Analysis. The International Journal of Logistics Management. 35. 10.1108/IJLM-09-2022-0362.

8.Barros, Rhaian & Filho, Jorge & Neto, Joao & Nascimento, Tiago. (2020). An Open-Design Warehouse Mobile Robot. 1-6. 10.1109/LARS/SBR/WRE51543.2020.9307137.