Implementing AI and Robotics for Baggage Transportation at Small and Medium-Sized Airports

S G N Sabir

Department of Applied Computer Science
UiT The Arctic University of Norway
Email: ssa219@uit.no

***Abstract*—This paper describes how I developed and implemented an AI-based robotic system to improve baggage transportation between terminals and planes at small and medium airports. The project focuses on creating autonomous robots that can navigate complex environments, handle tasks in a specific order, manage their energy levels, and work together to avoid conflicts and collisions. By using Reinforcement Learning (RL), specifically the Deep Deterministic Policy Gradient (DDPG) algorithm, and principles of multi-agent systems (MAS), the robots learn how to navigate efficiently and cooperate with each other. The paper explains the state representation, reward mechanisms, energy management strategies, and negotiation protocols used. The results show that the robots can learn efficient paths, follow the correct task sequence, consider their battery levels, and collaborate effectively, which enhances operational efficiency at airports.**

***Index Terms*—Artificial Intelligence, Robotics, Reinforcement Learning, Multi-Agent Systems, Airport Baggage Handling, Autonomous Navigation.**

I. Introduction

A. Background and Motivation

Airports are busy places where efficiency and coordination are very important, especially in handling baggage. Small and medium-sized airports often have limited resources and space, making it hard to optimize these operations. To address this, I started a project to introduce AI and robotics into the baggage transportation process, aiming to improve efficiency, reduce human error, and enhance safety.

The main idea is to develop autonomous robots that handle the transportation of baggage between the terminal and different planes. These robots need to fetch carts loaded with baggage, navigate the airport area, and perform tasks in a specific order. The environment is complex, with obstacles like buildings and parking areas, narrow passages that only allow one robot at a time, and moving elements such as other robots and objects.

B. Problem Statement

The project tackles several key challenges:

- **Navigation in Complex Environments**: Robots must learn to navigate efficiently while avoiding static and moving obstacles.
- **Task Sequencing**: Robots need to perform tasks in a specific order, like picking up carts before loading or unloading baggage.
- **Energy Management**: Robots must monitor their battery levels to ensure they have enough energy to complete tasks and return to charging stations.
- **Multi-Agent Coordination**: Multiple robots must work together to optimize operations, avoid conflicts, and prevent congestion in shared spaces.
- **Conflict Resolution**: Implementing negotiation methods to resolve conflicts when multiple robots aim for the same resource or path.

II. Related Work

A. Automated Guided Vehicles (AGVs) in Airports

Automated guided vehicles (AGVs) are crucial for efficient operations in airports, where they enhance the movement and handling of baggage and cargo. The design and control of these systems have been explored in-depth by several researchers. Le-Anh and De Koster (2004) present a comprehensive review of automated guided vehicle systems, addressing critical aspects such as guide-path design, vehicle scheduling, and battery management. They classify existing models and propose a framework that identifies key issues such as idle-vehicle positioning, a frequently overlooked area in AGV systems. Moreover, their research offers valuable insights into optimizing vehicle dispatching and deadlock resolution, which are essential for avoiding congestion in dynamic airport environments. The paper provides a useful basis for developing more advanced and autonomous AGVs, suggesting that further research should explore innovative scheduling algorithms and more sophisticated battery management techniques to improve efficiency in high-demand settings like airports [1].

B. Automated Guided Vehicles (AGVs) and Autonomous mobile robots in construction

Automated guided vehicles (AGVs) and autonomous mobile robots (AMRs) are becoming increasingly popular for solving a range of challenges in industries like logistics and manufacturing, as well as in high-risk jobs. Zhang et al.

(2023) explore the growing application of AGVs and AMRs in civil engineering, particularly in areas like road pavements, bridges, and general construction. Their paper reviews the recent advancements in these systems, focusing on visual tracking and navigation technologies. They compare different navigation techniques used in AGVs and AMRs, highlighting how new tracking integration technologies have improved the accuracy, reliability, and real-time performance of these vehicles. The authors also show how these robots are being used for tasks like defect detection and condition inspection, making construction more automated and less dependent on human labor. This work is particularly interesting because it points out the potential for AGVs and AMRs to transform how civil engineering projects are managed, helping to make the industry more efficient and safer [2].

C. Reinforcement Learning for Navigation

Lillicrap et al. [3] introduced the DDPG algorithm, which allows agents to learn optimal control policies in continuous action spaces. This algorithm's ability to handle complex action spaces made it suitable for my project's navigation challenges.

D. Multi-Agent Systems in Robotics

Busoniu et al. [4] provide a comprehensive survey of multi-agent reinforcement learning, highlighting the importance of coordination among agents to achieve common goals. Their work influenced my approach to implementing cooperation between multiple robots.

E. Energy Management in Autonomous Robots

Kim et al. [5] emphasize the need to include energy constraints in autonomous robot navigation to ensure ongoing operations and efficiency. This insight was crucial for integrating battery level considerations into my robots' decision-making processes.

F. Negotiation and Conflict Resolution

Kraus [6] explores negotiation strategies in multi-agent environments, detailing how agents can resolve conflicts and cooperate effectively. This literature guided the development of negotiation methods in my project.

G. Comparison with Current Project

My project builds upon these studies by integrating RL, MAS, and energy management into a unified system designed for airport baggage handling. Unlike traditional AGVs, the robots in my project learn and adapt to changing environments and can negotiate and cooperate with other robots in real-time.

III. Methodology

A. Solution Strategy

To address the challenges, I adopted a multi-step approach:

1) **Define an Appropriate State Representation**: Capturing all relevant information the robots need to make good decisions.
2) **Implement Reinforcement Learning (DDPG)**: Enabling robots to learn optimal navigation and task execution policies.
3) **Design Reward Mechanisms**: Guiding the robots toward desired behaviors through carefully designed rewards and penalties.
4) **Integrate Energy Management**: Ensuring robots consider their battery levels during planning and execution.
5) **Implement Multi-Agent Coordination and Negotiation**: Allowing robots to work together and resolve conflicts effectively.

B. Implementation Challenges and Mitigation

During the implementation, I faced several challenges and took steps to overcome them:

- **Complexity of the Environment**: Simulating a realistic airport environment with dynamic obstacles and multiple robots was computationally intensive. To mitigate this, I optimized the code by reducing unnecessary computations and using efficient data structures, which helped maintain real-time performance.
- **Hyperparameter Tuning**: Finding the right hyperparameters for the DDPG algorithm was challenging. Initial settings led to unstable learning or slow convergence. I mitigated this by conducting systematic experiments, adjusting one parameter at a time, and using guidance from existing literature to find suitable values.
- **Reward Function Design**: Balancing the reward function to encourage desired behaviors without unintended consequences was difficult. For example, too high a penalty for collisions made the robots overly cautious. I addressed this by incrementally adjusting the reward values and observing the robots' behaviors, aiming for a balance that promoted efficiency while avoiding negative actions.
- **Integration of Energy Management**: Incorporating battery level considerations into the robots' decision-making processes added complexity. Sometimes robots returned to charging stations at inappropriate times. To fix this, I adjusted the battery threshold levels and added penalties for abandoning tasks unnecessarily, ensuring that robots only returned to charge when necessary.
- **Multi-Agent Coordination**: Implementing effective communication and negotiation between robots was challenging. Initially, robots either ignored each other or became too cautious. I improved this by developing simple yet effective communication protocols and testing them extensively to ensure that robots could negotiate and make decisions that benefited the group.
- **Software Debugging and Testing**: Debugging the simulation with multiple interacting components was time-consuming. To mitigate this, I used logging extensively and tested individual components separately before integrating them, which helped identify and fix issues more efficiently.

C. Algorithms Used

1) Deep Deterministic Policy Gradient (DDPG): I chose the DDPG algorithm [3] because it effectively handles continuous action spaces, which is essential for smooth robot navigation. DDPG combines the benefits of Deep Q-Networks (DQN) [7] with actor-critic methods, allowing the robots to learn policies that map states to continuous actions.

Algorithm 1 DDPG Training Algorithm

Initialize actor network $\mu(s|\theta^\mu)$ and critic network $Q(s, a|\theta^Q)$ with random weights
Initialize target networks $\theta^{\mu'} \leftarrow \theta^\mu$, $\theta^{Q'} \leftarrow \theta^Q$
Initialize replay buffer R
for episode = 1 to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state s_1
  for t = 1 to T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$
    Execute action a_t, observe reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss:

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

    Update actor using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

    Set $s_t \leftarrow s_{t+1}$
  end for
end for

2) A Pathfinding Algorithm:* For initial path planning, I implemented the A* algorithm because it is efficient and accurate in finding the shortest path while considering obstacles.

Algorithm 2 A* Pathfinding Algorithm

Initialize the open list with the start node
Initialize the closed list as empty
while open list is not empty **do**
  Set current node to the node in open list with the lowest $f = g + h$
  if current node is the goal **then**
    Return the path by backtracking from the goal to start
  end if
  Move current node from open list to closed list
  for each neighbor of current node **do**
    if neighbor is not traversable or in closed list **then**
      Skip to the next neighbor
    end if
    Calculate tentative g score
    if neighbor not in open list or tentative g score is lower **then**
      Set neighbor's parent to current node
      Update neighbor's g and h scores
      if neighbor not in open list **then**
        Add neighbor to open list
      end if
    end if
  end for
end while
Return failure (no path found)

D. State Representation

The state vector is an array that captures:

1) **Current Position** (x, y): The robot's coordinates on the map.
2) **Battery Level** (soc): The robot's remaining battery percentage.
3) **Carrying Cart** (carry_cart): A value indicating whether the robot has picked up a cart.
4) **Target Position** (target_x, target_y): Coordinates of the current goal.
5) **Distance to Target**: The straight-line distance between the robot and the target.
6) **Exploration Rate** (ϵ): Balances exploring new actions and exploiting known good actions during learning.

Justification: This state representation provides the robot with essential information about itself, its goals, the environment, and its learning strategy. It includes both the robot's internal state (like battery level, whether it's carrying a cart) and external environment (like position, target location), enabling it to learn effective policies for navigation and task execution.

E. Reward Mechanism

Designing the reward function was crucial to guide the learning process. I assigned:

- **Positive Rewards**:
  - **Reaching the Target**: +500 (to encourage goal completion).

  - **Returning to Charging Station**: +100.
- **Negative Rewards**:
  - **Collision with Obstacles or Robots**: -700 (to strongly discourage collisions).
  - **Going Out of Bounds**: -1000 (to prevent leaving the operational area).
  - **Incorrect Task Sequence**: -500 (e.g., arriving at a plane without a cart).
- **Step Cost**: -1 per move to encourage efficiency.
- **Distance-Based Reward**: +20 times the decrease in distance to the target, rewarding progress toward the goal.

Justification: The reward function needed to balance encouraging the robots to complete tasks efficiently while penalizing undesirable behaviors like collisions or not following the task sequence. By including distance-based rewards, I ensured that the robots are continuously motivated to move closer to their targets.

F. Implementation Details

1) DDPG Agent: The DDPG agent includes:

- **Actor Network**: Decides the action to take given the current state.
- **Critic Network**: Estimates the value of the state-action pair.
- **Experience Replay Buffer**: Stores experiences for training to break correlations between samples.

Hyperparameters:

- **Learning Rates**: Actor (0.0001), Critic (0.001).
- **Discount Factor** (γ): 0.99.
- **Soft Update Rate** (τ): 0.001.
- **Replay Buffer Size**: 10,000.
- **Batch Size**: 64.

Justification: These parameters were chosen based on standard practices in RL, providing a balance between learning speed and stability.

2) Actor and Critic Networks: Both networks are implemented using fully connected neural networks with two hidden layers of 128 neurons each and ReLU activation functions. This setup provides enough capacity to model the complexities of the environment.

3) Energy Management: Robots reduce their battery level by 1 unit with each move. When the battery level falls below a threshold (e.g., 20%), the robot prioritizes returning to the charging station.

Challenges and Mitigation: Integrating energy management was challenging because robots sometimes chose to return to the charging station at inappropriate times, disrupting tasks. To mitigate this, I adjusted the threshold levels and modified the decision-making process to consider task urgency and remaining battery life, ensuring robots only returned to charge when necessary.

4) Task Sequencing and Cart Handling: Robots must pick up a cart before proceeding to a plane:

- **Carrying Cart Flag**: The variable indicates whether the robot has a cart.
- **Conditional Target Assignment**: The robot's target depends on whether it is carrying a cart.
- **Task Sequence Logic**:

Algorithm 3 Task Sequence Logic

```
if not carrying_cart then
    target ← cart_station
else if carrying_cart and not at_plane then
    target ← assigned_plane
else if at_plane then
    target ← terminal_loading_zone
end if
```

Challenges and Mitigation: Ensuring that robots strictly followed the task sequence required careful design of the reward function and state transitions. I mitigated this by adding penalties for incorrect task sequences and reinforcing correct sequences through positive rewards, which encouraged robots to follow the proper order.

5) Multi-Agent Coordination: Robots share information and coordinate to avoid conflicts:

- **Shared Memory**: Stores information about obstacles, failed paths, and other robots' positions.
- **Pheromone Map**: Robots leave and detect signals to indicate good or bad paths.
- **Collision Avoidance**: Robots check for potential collisions before moving and adjust paths as needed.
- **Negotiation Mechanisms**: Robots negotiate tasks based on battery levels, proximity to targets, and task urgency.

Challenges and Mitigation: Implementing effective communication was difficult. Initially, robots did not coordinate well, leading to conflicts. To mitigate this, I simplified the communication protocols and used basic negotiation strategies that allowed robots to make quick decisions without complex computations.

G. Meta-Parameters

- **Grid Size**: 20 units (divides the environment for movement and planning).
- **Robot Dimensions**: Width and height of 20 units.
- **Exploration Rate**: Starts at 0.5 and adjusts based on the robot's performance.

IV. Results

A. Simulation Setup

I developed a simulation environment using Pygame, representing the airport layout with obstacles, terminals, planes, carts, and charging stations. The robots are shown as colored rectangles (e.g., red, green, blue), and moving objects simulate dynamic obstacles.

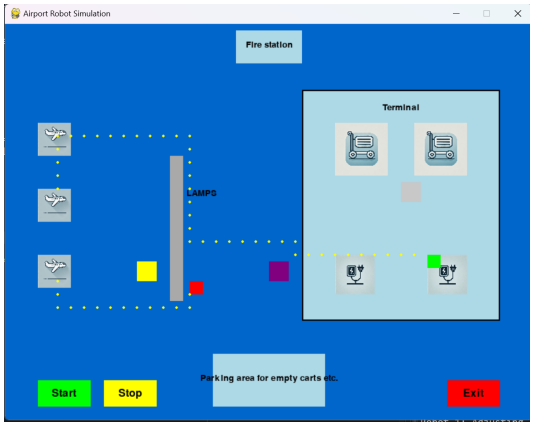

Fig. 1: GUI of the Airport Robot Simulation

B. Learning Progress

Fig. 2: Cumulative Rewards per Episode for Each Robot

1) Rewards per Episode: The cumulative rewards per episode for each robot show an upward trend, indicating that the robots are learning to maximize their rewards by completing tasks efficiently and avoiding penalties.

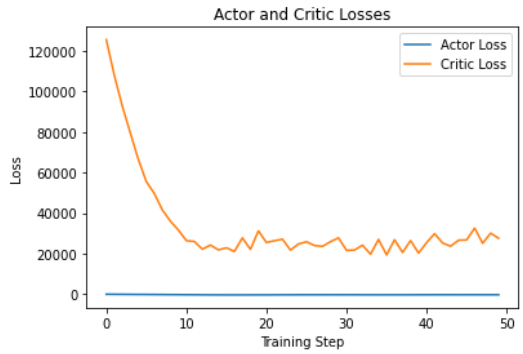

Fig. 3: Actor and Critic Losses During Training

2) Actor and Critic Losses: The losses for both the actor and critic networks decrease over time, showing that the neural networks are learning effective policies and value estimations.

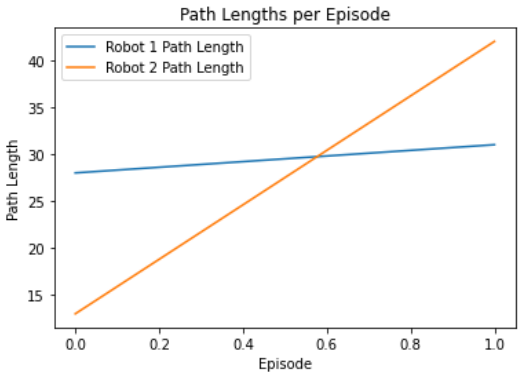

Fig. 4: Path Lengths Taken by the Robots Over Episodes

3) Path Lengths per Episode: The path lengths taken by the robots decrease as training progresses, suggesting that the robots are finding more efficient routes to their targets.

C. Behavioral Observations

- **Task Sequencing**: Robots consistently pick up carts before going to planes and return to the terminal after unloading. This indicates that the reward mechanism effectively enforces the correct task sequence.
- **Energy Management**: Robots monitor their battery levels and return to charging stations when necessary, ensuring they don't run out of power during a task. This behavior confirms that battery considerations are successfully integrated into decision-making.
- **Collision Avoidance**: Robots detect potential collisions with obstacles and other robots, adjusting their paths accordingly. The use of shared memory and signals helps in this process.
- **Cooperation and Negotiation**: When conflicts arise (e.g., multiple robots aiming for the same plane), robots negotiate based on battery levels and proximity, reassigning targets as needed. This demonstrates effective multi-agent coordination.

D. Challenges Encountered and Mitigation

Despite the successes, several challenges were encountered during the implementation and testing phases:

- **Computational Resources**: Training the robots required significant computational power and time. To mitigate this, I optimized the code by using efficient algorithms and, when possible, ran simulations on more powerful hardware.
- **Convergence Issues**: Sometimes, the learning process converged to less optimal behaviors. I addressed this by adjusting the learning rates and exploration strategies, and by providing more training episodes to allow the robots to explore better policies.
- **Dynamic Obstacles**: Including moving obstacles introduced unpredictability. Robots occasionally failed to adapt quickly, leading to collisions. To improve this, I enhanced the robots' sensing capabilities in the simulation and adjusted the collision avoidance algorithms.

- **Debugging Multi-Agent Interactions**: Finding issues in how robots interacted was complex. I mitigated this by isolating interactions and testing them individually, and by adding detailed logging to track the robots' decisions.
- **Balancing Exploration and Exploitation**: Robots sometimes explored too much or exploited known paths too early. I implemented an adaptive exploration rate that decreased over time, allowing the robots to balance exploration and exploitation more effectively.
- **Software Integration**: Integrating various components like the RL algorithm, simulation environment, and communication protocols was challenging. I managed this by modularizing the code, thoroughly testing each component before integration, and ensuring clear interfaces between modules.

V. Discussion

The results show that the robots effectively learn to navigate the airport environment while following operational constraints.

A. Successes

- **Efficient Navigation**: The robots learned to find optimal paths, reducing travel time and energy use. The decreasing trend in path lengths supports this observation.
- **Task Sequencing Compliance**: The reward structure effectively enforced the correct task sequence, with robots rarely attempting to skip steps. This compliance is crucial for smooth operations.
- **Energy Management**: Battery level considerations led to proactive recharging behaviors, preventing task failures due to depleted batteries. This reflects real-world operational needs.
- **Multi-Agent Cooperation**: Robots successfully coordinated to serve multiple planes without significant delays or conflicts. The negotiation mechanisms proved effective in resolving potential conflicts.

B. Limitations

- **Scalability**: The current implementation handles three robots; scaling up may require optimization to manage computational complexity. As the number of robots increases, the coordination overhead may become significant.
- **Real-World Applicability**: Simulations simplify certain aspects (like precise obstacle detection), and real-world deployment would need to address sensor noise, hardware limitations, and unexpected environmental changes.
- **Negotiation Complexity**: The negotiation mechanisms are relatively simple; more sophisticated protocols could enhance cooperation, especially in more complex scenarios.

C. Potential Improvements

- **Dynamic Exploration Rate**: Implementing an adaptive exploration strategy could improve learning efficiency, allowing robots to explore more when necessary and exploit known good behaviors when appropriate.
- **Advanced Negotiation Strategies**: Including auction-based or contract-net protocols for task allocation could enhance cooperation and efficiency.
- **Obstacle Prediction**: Improving the robots' ability to predict the movement of dynamic obstacles could improve navigation and safety.

VI. Conclusion

The project successfully demonstrates the use of AI and robotics to enhance baggage transportation at small and medium-sized airports. By integrating RL, MAS principles, and energy management strategies, the robots learned to navigate complex environments, perform tasks in the correct order, manage their battery levels, and cooperate effectively.

This approach has the potential to improve operational efficiency, reduce costs, and enhance safety in airport operations. The findings suggest that with further development and testing, such systems could be practical in real-world airport environments.

A. Future Work

- **Multi-Agent System with 3 Robots**: Currently the simulator is implemented for 2 robots, in future I will implement the simulator for 3 robots.
- **Robot power management**: Currently robots battery draining and to recharge themselves move to charging station mechanism is not implemented, In future I will implement this feature.
- **Robot task sequencing**: Currently robot is not starting from parking area, picking cart to the plane and returning to the terminal and not ending its episode in parking area. In the next version of this simulator, I will implement this task sequencing.
- **Real-World Testing**: Implementing the system with physical robots in a controlled environment to validate the simulation results. This would involve dealing with hardware integration, sensor reliability, and environmental variability.
- **Enhanced Learning Algorithms**: Exploring other RL algorithms, such as Proximal Policy Optimization (PPO), which may offer better convergence properties or be more robust to parameter settings in dynamic environment like airport.
- **Human-Robot Interaction**: Developing interfaces for human operators to oversee and intervene when necessary, ensuring that the system can integrate smoothly with existing airport operations.

References

[1] Le-Anh, Tuan & De Koster, René. (2004). A review of design and control of automated guided vehicle systems. European Journal of Operational Research. 171. 1-23. 10.1016/j.ejor.2005.01.036.

[2] Jianqi Zhang, Xu Yang, Wei Wang, Jinchao Guan, Ling Ding, Vincent C.S. Lee, "Automated guided vehicles and autonomous mobile robots for recognition and tracking in civil engineering," *Automation in Construction*, vol. 146, 2023, pp. 104699. Available: https://doi.org/10.1016/j.autcon.2022.104699.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[5] K. Kim, J. Kim, and J. Lee, "Energy-efficient autonomous robot navigation," *Robotics and Autonomous Systems*, vol. 62, no. 1, pp. 41–53, 2014.

[6] S. Kraus, "Negotiation and cooperation in multi-agent environments," *Artificial Intelligence*, vol. 94, no. 1-2, pp. 79–97, 1997.

[7] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.