RoboNavSim: A 3D Autonomous Navigation Simulation Using Multimodal AI Techniques

Technical Report – Capstone Project
Course Name: ITAI 2277 – Applied AI & Robotics

Instructor Name: Anna Rachapudi

Semester: Fall 2025

Date: December 10, 2025

Abstract

This technical report introduces RoboNavSim, a 3D robotic navigation simulation developed as the final capstone project for ITAI 2277 – Applied AI & Robotics. The system incorporates various artificial intelligence techniques, including heuristic navigation, sensor-based local obstacle avoidance, classical global path planning through A* search, and reinforcement learning utilizing Q-learning. The project's objective is to simulate and evaluate the performance of different AI navigation strategies within the same environment and constraints. Utilizing Python and Google Colab, RoboNavSim illustrates how robots can navigate intelligently in a 3D space, avoid obstacles, determine optimal paths, and acquire autonomous behaviors through experience. The project emphasizes essential robotics principles such as perception, decision-making, planning, and learning.

---

1. Introduction

Autonomous navigation represents a fundamental challenge in the fields of robotics and artificial intelligence. Robots functioning in unknown or changing environments need to detect obstacles, devise safe paths, adjust to uncertainties, and enhance their movements towards a target. This capstone project aims to create a simulation that illustrates how various AI techniques tackle the navigation issue.

RoboNavSim has been created as a 3D grid-based simulator that enables a virtual robot to navigate through an environment filled with obstacles. The robot is outfitted with simulated sensory capabilities and a variety of navigation algorithms. By combining reactive navigation, classical planning, and reinforcement learning, the system offers a cohesive platform for evaluating the advantages and disadvantages of each method. The simulation is fully implemented in Google Colab, guaranteeing reproducibility and ease of access.

This report documents the system architecture, algorithms implemented, design choices, experimental visualizations, and results.

---

2. System Overview

RoboNavSim functions within a 20 × 20 × 5 3D grid environment. This setting includes:

- A robot depicted as a 3D cube
- Static obstacles shown as 3D blocks
- A designated target goal position
- A simulated LIDAR-style sensing system
- Various navigation modules

The implementation is done in Python utilizing:

- NumPy for numerical calculations
- Matplotlib for 3D visual representation
- Custom Python classes and functions to facilitate navigation behavior

The architecture is composed of:

1. Environment Module – responsible for grid creation, obstacle arrangement, and rendering

2. Robot Module – oversees robot state, sensing capabilities, and movement

3. Navigation Module – includes the following algorithms:

o Greedy heuristic

o Sensor-based obstacle avoidance

o A* path planning

o Q-learning reinforcement learning

4. Visualization Module – generates 3D plots, policy maps, and value heatmaps

This modular design enables the independent testing of each navigation strategy under consistent conditions.

---

3. Approaches and Techniques

3.1 Local Heuristic Navigation

The heuristic system guides the robot towards its target by choosing the move that reduces the Euclidean distance.

This approach does not take obstacles into account until they are detected, which makes it straightforward yet constrained.

Strengths: Quick, lightweight

Weaknesses: Can become trapped in clusters of obstacles

3.2 Sensor-Based Reactive Navigation

The robot emulates proximity sensing within a certain radius. All unsafe maneuvers near obstacles are eliminated. Gaussian noise is introduced to movement to mimic actuator flaws.

Behavior:

- Prevents collisions
- Maintains a local safety buffer
- Moves aggressively when feasible

This method mirrors real-world robotics where the robot must respond based on limited sensor data.

3.3 A Global Path Planning*

A* views the environment as a grid graph and calculates an optimal route:

- g(n) = cost from start to current
- h(n) = heuristic distance to the target
- f(n) = g(n) + h(n)

A* provides a globally efficient path that circumvents all obstacles.

Strengths: Optimal route when the complete map is available

Weaknesses: Not ideal for unfamiliar environments

A 3D visualization illustrates the entire planned path.

3.4 Q-Learning Reinforcement Learning Agent

The RL agent represents navigation as a Markov Decision Process (MDP):

- States: (x, y) grid positions
- Actions: 8 possible moves (including diagonals)
- Rewards:
  - +100 for reaching goal
  - –10 for colliding with an obstacle
  - –5 for boundary collisions
  - –1 per time step

The Q-table has shape:
(grid_size, grid_size, 8)

Training uses ε-greedy exploration with decay. After training, a greedy policy is extracted and used to generate the learned path.

---

4. Experiments and Results

4.1 Path Comparisons

Three navigation strategies were tested:

Method	Path Quality	Obstacle Avoidance	Knowledge	Notes
Greedy	Poor/Fair	Weak	Local only	Fast but easily trapped
Sensor-Based	Fair	Strong	Local	Good real-robot simulation
A*	Excellent	Perfect	Global	Requires full map
Q-Learning	Good	Strong	Learned	Improves over episodes

4.2 Visual Outputs

The simulation generates:

- 3D environment rendering
- Reactive path visualization
- A* global path overlay
- RL learned path
- Reward curve showing training progress
- Value function heatmap
- Policy arrow map

These visualizations illustrate the reasoning of each navigation method.

---

5. Analysis

A* vs RL

The heuristic system guides the robot towards its target by choosing the move that reduces the Euclidean distance.

This approach does not take obstacles into account until they are detected, which makes it straightforward yet constrained.

Strengths: Quick, lightweight

Weaknesses: Can become trapped in clusters of obstacles

3.2 Sensor-Based Reactive Navigation

The robot emulates proximity sensing within a certain radius. All unsafe maneuvers near obstacles are eliminated. Gaussian noise is introduced to movement to mimic actuator flaws.

Behavior:

- Prevents collisions
- Maintains a local safety buffer
- Moves aggressively when feasible

This method mirrors real-world robotics where the robot must respond based on limited sensor data.

3.3 A Global Path Planning*

A* views the environment as a grid graph and calculates an optimal route:

- g(n) = cost from start to current
- h(n) = heuristic distance to the target
- f(n) = g(n) + h(n)

A* provides a globally efficient path that circumvents all obstacles.

Strengths: Optimal route when the complete map is available

Weaknesses: Not ideal for unfamiliar environments

A 3D visualization illustrates the entire planned path.

3.4 Q-Learning Reinforcement Learning Agent

The RL agent represents navigation as a Markov Decision Process (MDP):

- States: (x, y) grid positions

- Classical AI planning
- Reinforcement learning
- Python simulation development
- Debugging large multi-stage notebooks

---

7. Future Work

To enhance RoboNavSim:

- Implement Deep Q-Learning (DQN) with neural networks
- Add dynamic obstacles and multi-agent simulation
- Integrate with PyBullet or ROS for full physics simulation
- Add velocity, acceleration, and rotation modeling

- Expand to 3D movement rather than only 2.5D navigation

---

8. Conclusion

RoboNavSim effectively showcases various artificial intelligence techniques applied in robotic navigation.

By merging heuristic control, sensor-based responsiveness, global planning, and reinforcement learning, this project offers a thorough insight into AI-powered autonomous navigation. The system not only fulfills the capstone project requirements but also lays a scalable groundwork for future exploration in robotics and AI.

This initiative emphasizes the significance of integrating planning, perception, and learning within a cohesive simulation. The findings illustrate how distinct AI methodologies tackle the same navigation problem, each with its own unique advantages and drawbacks. Ultimately, RoboNavSim serves as an educational and visually engaging representation of intelligent robotic navigation within a simulated 3D environment.

Refences:

All source code, figures, and documentation are available in the GitHub repository

https://github.com/joelleyarro03/Capstone_RoboNavSim_WillianeYarro_ITAI2277.git

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.
— Referenced for heuristic search, agents, and A* path planning.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.
— Referenced for Q-learning reinforcement learning algorithms.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT Press.
— Referenced for robotic sensing, uncertainty, and local obstacle avoidance.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge University Press.
https://planning.cs.uiuc.edu
— Referenced for classical planning techniques including A*.

Matplotlib Development Team. (2024). *Matplotlib: Visualization with Python*.
https://matplotlib.org
— Used for 3D rendering and environment visualization.

NumPy Developers. (2024). *NumPy*. https://numpy.org
— Used for matrix operations and computation.

Google Research. (2023). *Google Colab*. https://colab.research.google.com
— Platform used to run and execute the simulation.

OpenAI. (2023). *AI principles and safety in robotics*. OpenAI Research Notes.
— Referenced for general AI safety principles relevant to learning agents.