CS 520

Introduction To Artificial Intelligence

Project 2: Space Rats

Group Members:

Abhishek Jani (aj1121)
Shrey Patel (sp2675)
Mustafa Adil (ma2398)

Contents

1 Introduction

In this project, we are working on a simulation where a bot is tasked with finding and capturing a ”space rat” inside a grid-based ship. The space rat can either stay in one place or move randomly to nearby cells, making the task more challenging. To solve this, we created two strategies: the **Baseline Bot** and **Our Bot**, and tested them under different conditions. These include changes in how reliable the bot's sensor is, controlled by a parameter called α. The goal is to create better strategies that help the bot find and catch the rat quickly and efficiently.

1.1 Primary Objectives

The primary objectives of this project are to achieve efficient pathfinding by minimizing the number of actions taken by the bot to capture the rat, to develop strategies that allow dynamic adaptation to both stationary and moving rat scenarios, and to compare and analyze the performance of the two bot strategies across a range of α values.

2 Problem Formulation and Project Objectives

2.1 Problem Formulation

The space rat problem is such that a bot needs to find a rat on a 30×30 grid using feedback from its sensors. This problem has several key components. The **state space** is the 30×30 ship grid, where each cell can either be open (the bot can move there) or blocked (the bot cannot move there). The **initial state** involves both the bot and the rat starting in random open cells on the grid. The **goal state** is achieved when the bot reaches the same cell as the rat, effectively capturing it.

The rat (if it moves) randomly moves to one of the nearby open cells at every step.

. The bot receives ”ping” signals that indicate the likelihood of the rat being nearby. This likelihood is calculated using the formula:

$$P(\text{Ping}) = e^{-\alpha \cdot (\text{distance}-1)}$$

In this formula, α determines how reliable the sensor is. A lower α value means the sensor is very sensitive and gives frequent pings, while a higher α value means the sensor is less sensitive but provides more precise pings.

2.2 Project Objectives

The goal of this project is to create a system that helps the bot find the rat efficiently. This involves representing the ship as a grid with unique distance signatures for each cell and using probabilities to guess where the rat might be. The project also focuses on designing efficient strategies for the bots. The Baseline Bot uses simple probability-based rules, while Our Bot uses A* pathfinding to plan the best path. Another aim is to handle changes in the environment by making the bots adapt to both stationary and moving rats, using a method called probability diffusion to guess where a moving rat might go. Finally, the project evaluates how well each bot performs by counting the number of actions they take and checking their accuracy under different sensor reliability levels (α).

3 Key Concepts and Formulas

In this section, we explain the main ideas and formulas used in the project. These are the key concepts that helped us design and build the bots and their strategies.

3.1 Digital Signatures for Knowledge Base

A **digital signature** is a way to uniquely describe each open cell in the grid. It shows how far the nearest obstacles (blocked cells) are in all four directions: **up, down, left, and right**.

Purpose of Digital Signatures

1. **Localization**: The bot uses the distances to nearby obstacles (up, down, left, right) to create a unique digital signature for its current position. It then compares this signature with a list of signatures for all open cells in the grid. If the sensed signature matches exactly one location, the bot can confidently identify its position.

   If multiple cells share the same signature, the bot cannot determine its exact location and considers all possibilities. When the bot gets stuck or finds that no progress is being made in narrowing down its location, it makes a random move in one of the four directions (up, down, left, right) to explore the grid further and potentially reduce ambiguity. This random movement helps the bot escape situations where it cannot differentiate between multiple possible locations.

2. **Reduction of Search Space**: The use of digital signatures helps the bot narrow down the possible locations it could occupy in the grid. Instead of considering the entire grid, it only focuses on cells with matching signatures, significantly reducing the search space and speeding up the localization process.

How It Works

For a cell (x, y):

- The bot calculates distances to the nearest blocked cells in each direction:
  - **Up**: Number of open cells between (x, y) and the nearest blocked cell above.
  - **Down**: Number of open cells to the nearest blocked cell below.
  - **Left**: Number of open cells to the nearest blocked cell on the left.
  - **Right**: Number of open cells to the nearest blocked cell on the right.

3.2 Sensor Feedback (Ping Probability)

The bot receives a **ping** signal that provides information about the likelihood of the rat being nearby. This probability is determined by an **exponential decay function**:

$$P(\text{Ping}) = e^{-\alpha\cdot(\text{distance}-1)}$$

Components of the Formula

1. **Distance**:
$$\text{distance} = |x_{\text{bot}} - x_{\text{rat}}| + |y_{\text{bot}} - y_{\text{rat}}|$$

2. **Alpha (α)**:
   - Sensor sensitivity parameter:
   - **Low** α: High sensitivity, frequent pings with broad coverage.
   - **High** α: Low sensitivity, fewer pings with precise coverage.

Behavior

- **Low** α ($\alpha \approx 0.1$):
  - Pings almost always return a signal.
  - Useful for initial localization but can overwhelm the bot with redundant information.
- **High** α ($\alpha \approx 0.9$):
  - Pings are rare but provide precise information when triggered.
  - Suitable for scenarios where precise localization is required.

Example

- Rat is at $(5,5)$, bot is at $(3,3)$. - **Distance**: $|5-3|+|5-3| = 4$ - **Ping Probability**:

- $\alpha = 0.1$: $P(\text{Ping}) = e^{-0.1\cdot(4-1)} = e^{-0.3} \approx 0.74$
- $\alpha = 0.9$: $P(\text{Ping}) = e^{-0.9\cdot(4-1)} = e^{-2.7} \approx 0.067$

3.3 Probability Grid

The **probability grid** is a 2D array that represents the likelihood of the rat being in each cell. It is updated dynamically based on:

1. **Sensor Feedback**:
   - If a ping is heard, increase probabilities for cells closer to the bot.
   - If no ping is heard, reduce probabilities for nearby cells.

2. **Rat Movement** (in moving scenarios):
   - Spread probabilities across neighboring open cells to account for the rat's random movement.

Update Rule

For each cell (x, y):

- **Ping Heard**:
$$P(x, y) \propto P_{\text{prev}}(x, y) \cdot e^{-\alpha \cdot (\text{distance to bot}-1)}$$
- **No Ping Heard**:
$$P(x, y) \propto P_{\text{prev}}(x, y) \cdot (1 - e^{-\alpha \cdot (\text{distance to bot}-1)})$$

3.4 Diffusion of Probability in Moving Rat Scenario

In the moving rat scenario, the bot updates the probability grid to account for where the rat might move. At each step, the probability of the rat being in a specific cell is spread out to its neighboring open cells. This is done using the formula:

$$P_{\text{new}}(x, y) = \sum_{\text{neighbors}} \frac{P_{\text{prev}}(x', y')}{\text{number of open neighbors}}$$

3.5 A* Pathfinding

Our Bot uses the A* algorithm for efficient pathfinding:

- **Cost Function**:
$$f(x) = g(x) + h(x)$$
where $g(x)$ is the cost to reach cell x and $h(x)$ is the heuristic (Manhattan distance to the target).
- **Efficient Handling of Obstacles**: Avoids blocked cells while moving toward the target.

Pseudo-code

A* Pathfinding Pseudo-code

```
Initialize open set with start position
While open set is not empty:
    Pop cell with lowest f(x)
    If cell is the target:
        Return path
    For each neighbor:
        If neighbor is open and not visited:
            Update g(x) and f(x)
            Add to open set
```

4 Implementation and Algorithms

This section details the step-by-step implementation of the key components of the project, including the environment setup, probability grid, and bot strategies. Each subsection includes a pseudo-code representation and algorithm description.

4.1 Ship Environment

The ship is represented as a 30x30 grid, with each cell either open or blocked. Border cells are always blocked, while interior cells have a 70% chance of being open.

Ship Initialization Algorithm

```
Initialize a 30x30 grid (all cells open by default)
For each cell (x, y) in the grid:
    If cell is on the border:
        Mark as blocked
    Else:
        Mark as blocked with a 30% probability
```

Border cells are used to ensure the bot remains within the grid, and blocked cells create obstacles, making pathfinding non-trivial.

4.2 Probability Grid and Knowledge Base

The probability grid shows how likely it is that the rat is in each open cell. The bot uses digital signatures in its knowledge base to figure out the possible locations of the rat.

Probability Grid Initialization Algorithm

```
Initialize probability grid:
    For each open cell (x, y):
        Set P(x, y) = 1 / (total open cells)
```

Probability Update Rule

- If a ping is heard:

$$P(x, y) \propto P_{\text{prev}}(x, y) \cdot e^{-\alpha \cdot (\text{distance to bot} - 1)}$$

- If no ping is heard:

$$P(x, y) \propto P_{\text{prev}}(x, y) \cdot (1 - e^{-\alpha \cdot (\text{distance to bot} - 1)})$$

After updating probabilities based on sensor feedback, the probabilities are normalized:

$$P(x, y) = \frac{P(x, y)}{\sum_{\text{all cells}} P(x, y)}$$

4.3 Baseline Bot

The Baseline Bot follows a straightforward strategy. It senses the rat using the sensor (ping), updates the probability grid based on the ping, and moves step-by-step toward the most probable cell.

Baseline Bot Algorithm

```
While the rat is not found:
    Sense ping
    Update probability grid
    Identify the cell with the highest probability
    Move step-by-step toward the target cell
    If bot reaches the rat's cell:
        End simulation
```

However, the Baseline Bot does not use the best paths, which makes it less efficient. It also has trouble with the moving rat scenario because it keeps recalculating probabilities without accounting for the rat's movement.

4.4 Our Bot

Our Bot is better than the Baseline Bot because it uses A* pathfinding to find the best paths. It also updates the probability grid to handle the rat's movement and uses probability diffusion to guess where the rat might go in the moving scenario.

Our Bot Algorithm

```
While the rat is not found:
    Sense ping
    Update probability grid
    Identify the cell with the highest probability
    Plan the shortest path to the target using A*
    Move along the planned path
    If the rat moves:
        Recalculate probabilities and re-plan path
    If bot reaches the rat's cell:
        End simulation
```

4.5 A* Pathfinding Algorithm

The A* Algorithm helps the bot find the shortest path to the target cell quickly. It uses a heuristic, which is the **Manhattan Distance** to the target, to guide its search.

A* Pathfinding Algorithm

```
Initialize open set with the bot's starting position
Initialize g_score and f_score for all cells:
    g_score =  (distance from start)
    f_score =  (estimated total cost)
Set g_score[start] = 0 and f_score[start] = h(start)
While open set is not empty:
    Pop cell with the lowest f_score
    If the cell is the target:
        Return the reconstructed path
    For each neighbor of the current cell:
        If neighbor is open:
            Calculate tentative_g = g_score[current] + 1
            If tentative_g < g_score[neighbor]:
                Update g_score and f_score
                Add neighbor to the open set
Return "No Path Found" if the open set is exhausted
```

4.6 Moving Rat Dynamics

In the moving rat scenario, the rat changes its position with each step. The bot updates its probability grid to spread the likelihood of the rat being in nearby cells.

Rat Movement Algorithm

```
For each timestep:
    Select a random direction (up, down, left, right)
    If the neighboring cell is open:
        Move the rat to the new cell
    Else:
        Keep the rat in the current cell
```

Probability Diffusion Algorithm

```
For each open cell (x, y):
    Set P_new(x, y) = sum(P_prev(neighbor) / number of open neighbors)
Normalize the probability grid
```

5 Scenarios and Bot Strategies

This section explains the different scenarios used to test the bots and describes how each bot works in both stationary and moving rat environments.

5.1 Stationary Rat Scenario

In this scenario, the rat stays in the same spot for the entire simulation. The bots use sensor feedback and probability updates to find and catch the rat.

Baseline Bot Strategy

The Baseline Bot updates the probability grid using the ping signal, finds the cell with the highest probability, and moves toward it step by step. It keeps recalculating probabilities at each step until it catches the rat. This bot works well when α is low because the sensor gives frequent and accurate pings. However, it often takes inefficient paths because it doesn't plan the best route and wastes moves in grids with many obstacles.

Our Bot Strategy

Our Bot uses A* pathfinding to plan the shortest path to the target. It makes better use of the probability grid by recalculating paths only when needed and focusing on cells with high probabilities. This bot takes far fewer actions than the Baseline Bot and avoids dead-end paths, making it better at handling grids with many obstacles. However, it depends on accurate sensor feedback, and its performance drops slightly when α is very high because the sensor gives fewer pings.

Performance (Stationary Rat)

Alpha (α)	Baseline Bot (Actions)	Our Bot (Actions)
0.1	152	62
0.5	632	236
0.9	808	254

Table 1: Performance comparison of Baseline Bot and Our Bot in the stationary rat scenario.

5.2 Moving Rat Scenario

In this scenario, the rat moves randomly to a nearby open cell at each step. The bots need to adjust to the rat's changing position by updating their knowledge base and probability grid.

Baseline Bot Strategy

The Baseline Bot recalculates probabilities after every move and goes step-by-step toward the cell with the highest probability. It doesn't consider that the rat might move, so it often targets outdated probability cells. The bot works okay when α is low, but it struggles in dynamic environments, wasting moves and missing chances to catch the rat.

Our Bot Strategy

Our Bot updates the probability grid to account for the rat's movement using a diffusion model. It recalculates paths smartly and uses A* pathfinding to avoid unnecessary moves.

This bot adapts well to the rat's movement, works efficiently across all α values, and handles grids with obstacles and dynamic targets much better than the Baseline Bot.

Performance (Moving Rat)

Alpha (α)	**Baseline Bot (Actions)**	**Our Bot (Actions)**
0.1	558	119
0.5	3313	227
0.9	3197	758

Table 2: Performance comparison of Baseline Bot and Our Bot in the moving rat scenario.

5.3 Key Observations

In the stationary rat scenario, both bots work better at low α values ($\alpha \leq 0.3$) because the sensor gives frequent feedback. Our Bot does much better than the Baseline Bot by taking fewer actions. In the moving rat scenario, the Baseline Bot struggles a lot because it can't adjust to the rat's movement, while Our Bot stays efficient and adaptable, even at high α values. When α is very high ($\alpha > 0.8$), both bots face difficulties because the sensor gives less reliable feedback, but Our Bot still performs better thanks to its smarter pathfinding and ability to adapt.

6 Results and Data Analysis

This section looks closely at how the bots performed in both stationary and moving rat scenarios at different levels of sensor sensitivity (α). It explains the main trends, compares the bots, and shares useful insights from the results.

6.1 Visualizing the Ship Layout and Bot Positions

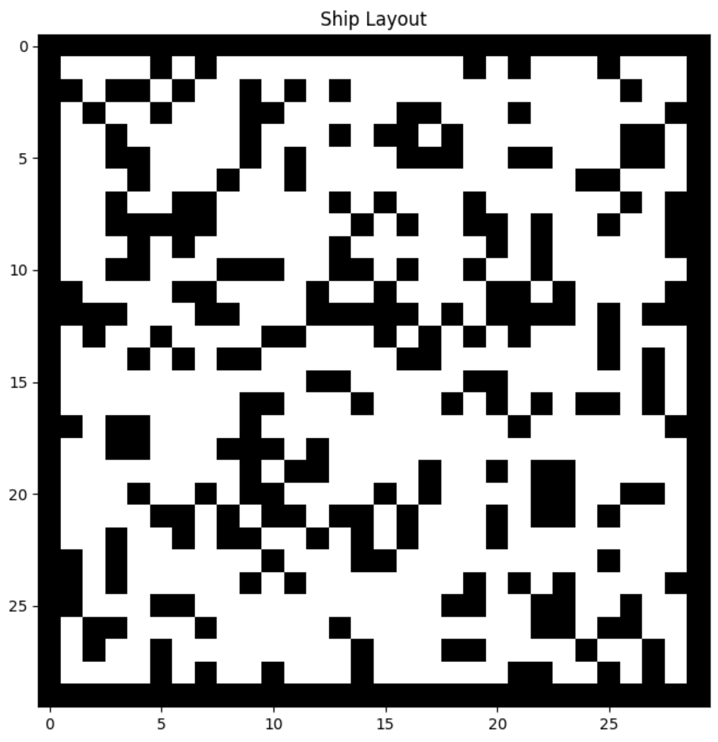

Figure 1: Ship Layout with Obstacles

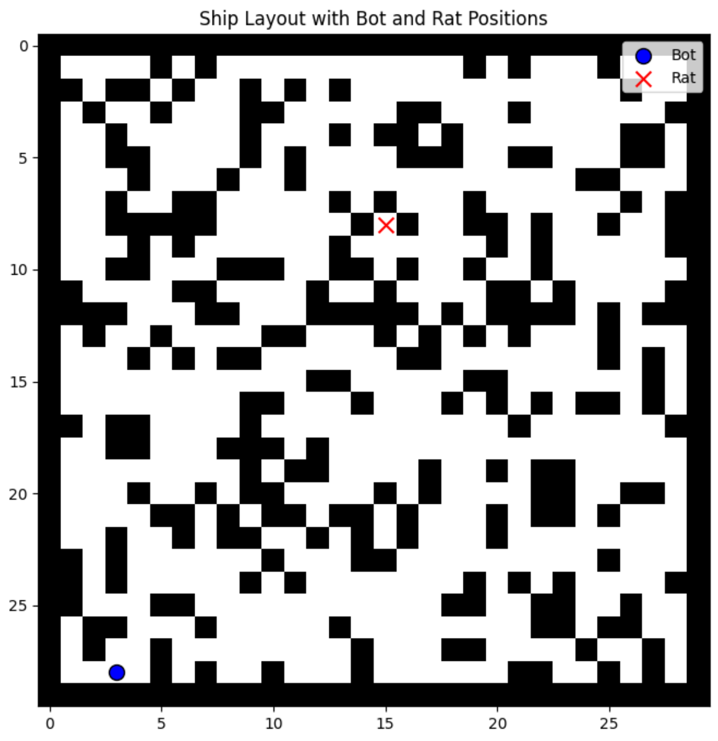

Figure 2: Ship Layout with Bot and Rat Positions

6.2 Performance Metrics

The primary performance metric is the number of actions taken by the bot to capture the rat. Fewer actions indicate higher efficiency and accuracy.

6.3 Results for Stationary Rat Scenario

Alpha (α)	Baseline Bot (Actions)	Our Bot (Actions)
0.1	152	62
0.2	164	100
0.3	223	112
0.4	540	122
0.5	632	236
0.6	976	232
0.7	1024	268
0.8	632	214
0.9	808	254

Table 3: Performance Comparison for Stationary Rat Scenario

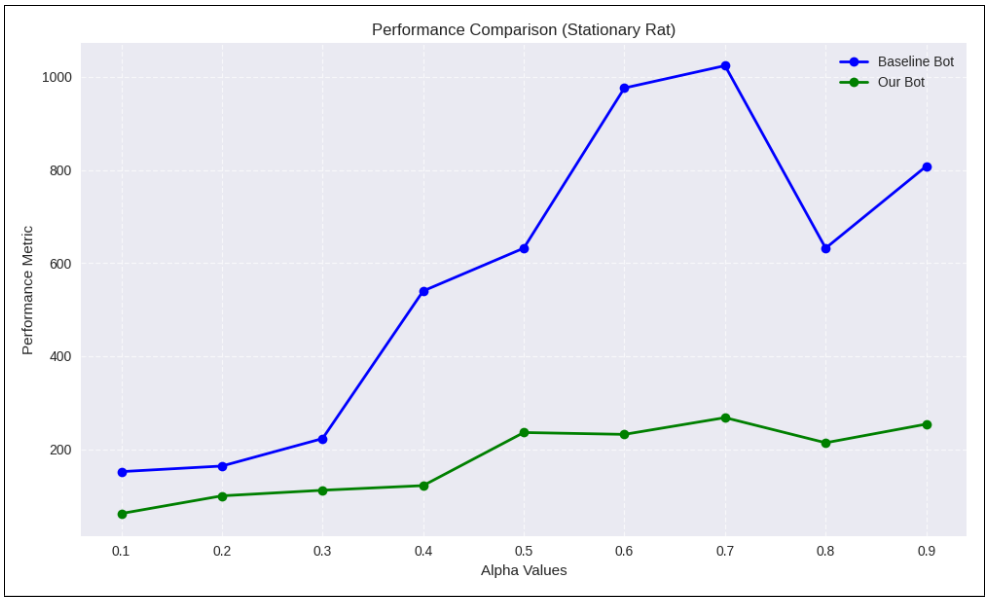

Figure 3: Performance Comparison of Baseline Bot vs Our Bot (Stationary Rat).

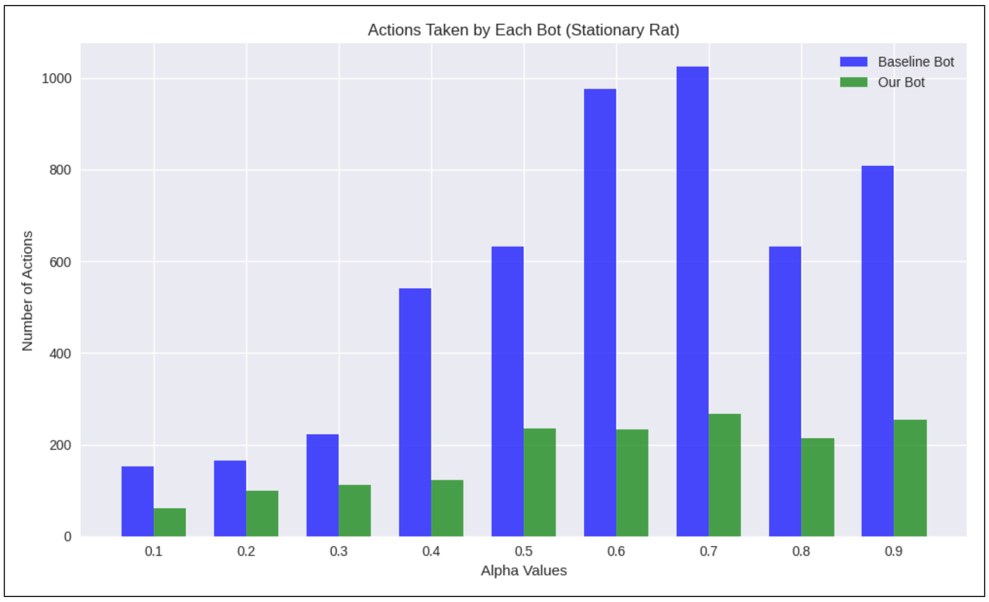

Figure 4: Actions Taken by Each Bot (Stationary Rat - Bar Plot).

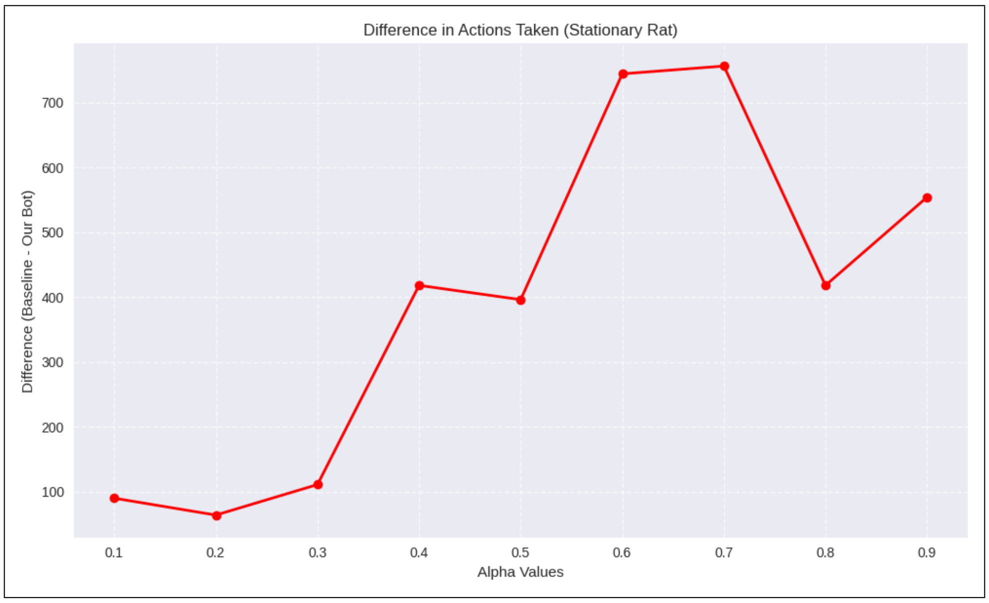

Figure 5: Difference in Actions Taken (Stationary Rat Scenario).

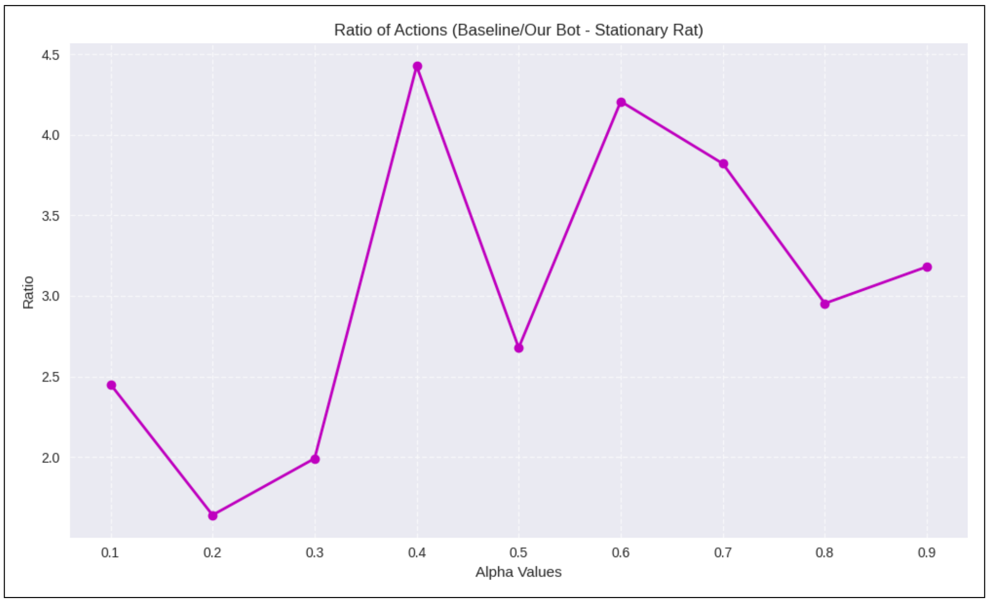

Figure 6: Ratio of Actions Taken (Baseline Bot to Our Bot - Stationary Rat).

6.4 Results for Moving Rat Scenario

Alpha (α)	**Baseline Bot (Actions)**	**Our Bot (Actions)**
0.1	558	119
0.2	658	152
0.3	973	254
0.4	2666	342
0.5	3313	227
0.6	3453	233
0.7	2912	439
0.8	3286	684
0.9	3197	758

Table 4: Performance Comparison for Moving Rat Scenario

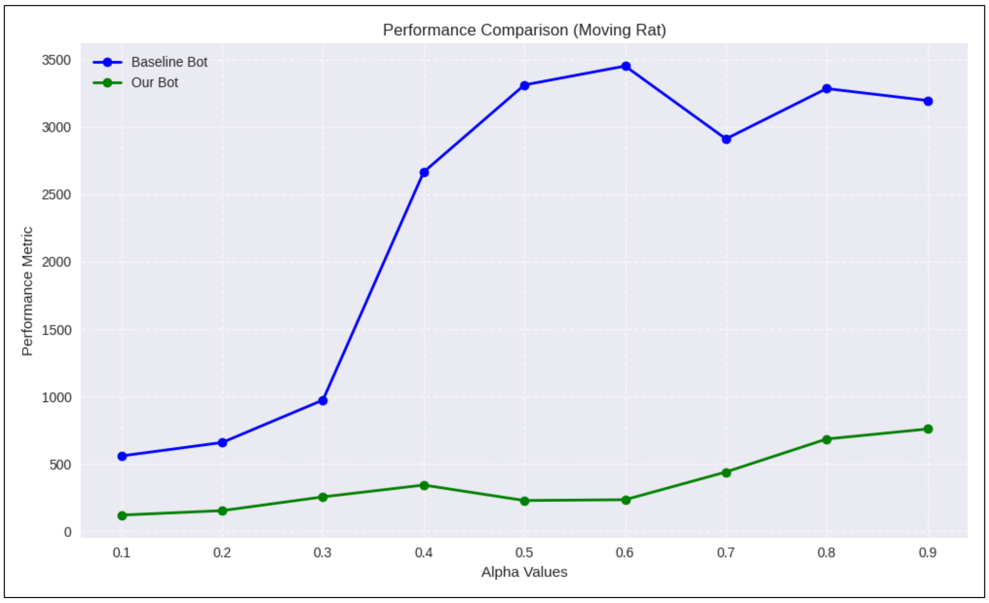

Figure 7: Performance Comparison of Baseline Bot vs Our Bot (Moving Rat).

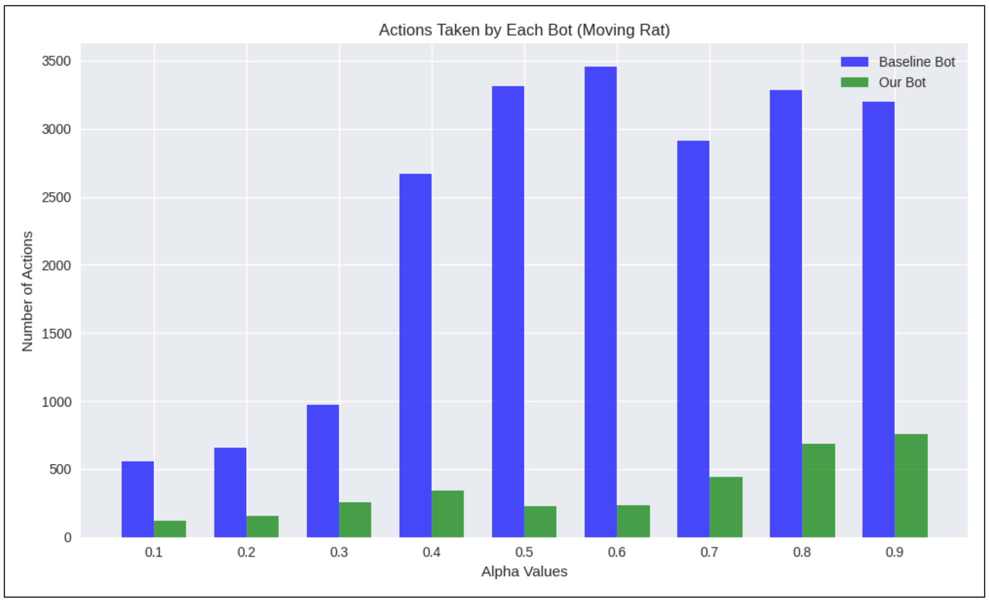

Figure 8: Actions Taken by Each Bot (Moving Rat - Bar Plot).

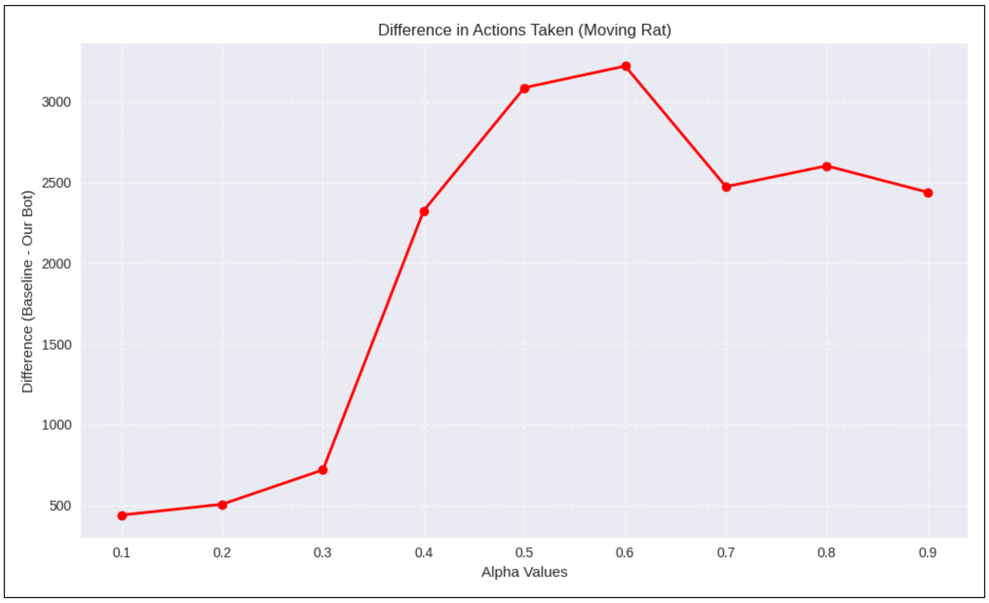

Figure 9: Difference in Actions Taken (Moving Rat Scenario).

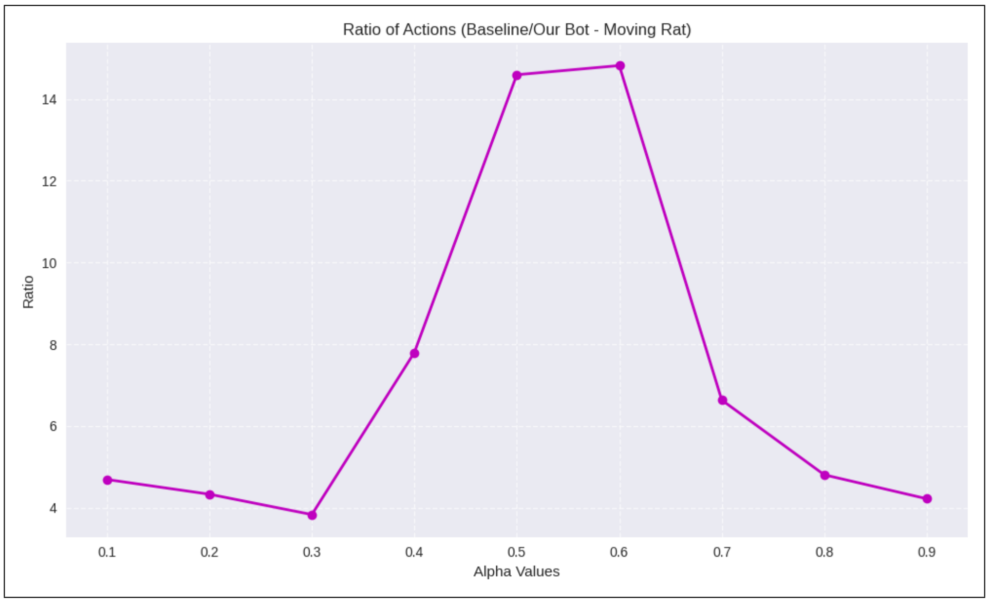

Figure 10: Ratio of Actions Taken (Baseline Bot to Our Bot - Moving Rat).

6.5 Key Observations

Our Bot does better than the Baseline Bot in the stationary rat scenario, needing fewer actions at all α values. In the moving rat scenario, Our Bot works well in dynamic environments and adjusts efficiently to the rat's movement. When α is higher, the Baseline Bot's performance gets much worse, but Our Bot stays reliable.

7 Failure Modes

For the Baseline Bot, in the stationary rat scenario, it often takes inefficient paths by moving step-by-step toward the cell with the highest probability. This causes too many recalculations and unnecessary movements. It also gets stuck in dead ends, wasting actions as it has to backtrack. In the moving rat scenario, the bot targets outdated probability cells because it cannot adjust its strategy to follow the rat's movement, which causes significant delays.

For Our Bot, in the stationary rat scenario, the A* pathfinding recalculations can make it slower in real-time situations. In the moving rat scenario, the probability diffusion can spread the chances too widely, making it harder to focus on the rat's actual location. At very high α values, the sensor gives less feedback, which makes the probability updates less reliable.

8 Conclusion

In this project, we designed, built, and tested two bot strategies—Baseline Bot and Our Bot—to solve the space rat problem in both stationary and moving environments. We used probability-based systems, sensor feedback, and advanced pathfinding to handle the challenges of working in unpredictable situations.

This project showed how useful knowledge-based systems can be. Using digital signatures and probability grids helped the bots find the rat more effectively. Our Bot was consistently better than the Baseline Bot because it used A* pathfinding and updated probabilities dynamically. Sensor sensitivity (α) affected performance, with both bots working best at low α values. However, Our Bot handled higher α values better. Adapting to changing situations was very important, and Our Bot managed this well by using probability diffusion and adjusting paths in real-time.

8.1 Team Contribution

- **Abhishek Jani:**
  - Led the implementation of the A* pathfinding algorithm and integrated it with probability updates for Our Bot.
  - Designed experiments to evaluate bot performance under varying α values.
- **Mustafa Adil:**
  - Contributed to developing the Baseline Bot strategy and implemented the sensor feedback mechanism.
  - Conducted initial grid simulations and supported data analysis for stationary rat scenarios.
- **Shrey Patel:**
  - Focused on dynamic rat movement modeling and probability diffusion updates for the moving rat scenario.
  - Analyzed failure modes and proposed optimizations to enhance bot adaptability.