**Project 2: Space Rate**

**Ship Layout:  
  
For the Ship layout (Grid) I have use different symbol for each thing.  
  
1. 1 - For the indicate block cell in ship**

**2. 0 - For the indicate open cell in ship**

****

**Updating Space Rat Knowledge Base (Stationary Rat)**

The rat knowledge base is a dictionary of open cells as keys and the values are probabilities of having a rat in that open cell. The rat knowledge base is initialized and updated using the following steps:

**Step 1: Initialization** At first, the rat knowledge base is initialized with all the open cells having an equal probability of having a rat. The probability distribution is initialized as follows:

Where Nopencells is the total number of the open cells.

**Step 2: Update space rat knowledge base based on rat found or not found** Then, the bot checks to see if there’s a rat in the bot’s current location. If the bot does not find the rat in it’s current location the probability of having a rat in that cell is set to 0 and the probability distribution of rat knowledge base is updated as follows:

For all Locations Li in L except Lbot :

**Step 3: Update space rat knowledge base based on ping or no ping**

The bot then uses the space rat detector to listen for a ping. Depending on whether a ping is heard or not, the probabilities in the rat knowledge base are updated using Bayesian inference as follows:

**If the bot hears a ping:**

**If the bot does not hear a ping:**

Here, we can see that probability of receiving a ping or not receiving a ping that the rat is in location Li is calculated using a

(As follow:

)

The normalization factor in terms of Bayesian inference, P (ping = 1) is calculated as follows:

Similarly, is calculated as follows:

The prior probability, is the probability of the rat being found in the cell which can be obtained from the space rat knowledge base.

Using the Bayesian inference the space rat knowledge base is updated based on whether a ping is received or not.

**Step 4:** Move towards the cell with the highest probability The bot starts moving towards the cell that has the highest probability in the space rat knowledge base.

The bot continues step 2, 3, and 4 until the rat is caught.

**Design and Algorithm: Baseline Bot**

As instructed, the baseline bot works in two phases:

Phase 1: Identify where the bot is

Phase 2: Track the Space rat

**Phase 1: Bot Localization**

1. **Initialize Knowledge Base (KB):** Create an Initial bot knowledge base with all open cells as possible locations for the bot.
2. **Sense Blocked Neighbors:** Sense how many of the eight neighboring cells are currently blocked.
3. **Rule out Location:** Update the bot knowledge base by ruling out location that do not match the sensed value.
4. **Find Common Direction**: Identify the direction that is most commonly open among the remining possible locations in the knowledge base. Select the direction that has the highest count of opening cells. However, if there are more than one such direction, the bot selects one at random.
5. **Attempt to Move:** Attempt to move to most common direction.
6. **Update Knowledge base based on attempted move**: If the move is successful, rule out all location where the move was blocked. If unsuccessful, rule out all the locations where the move was not blocked.

The bot keeps repeating step 2,3,4,5 and 6 until only one possible location remains in knowledge base which is the bot location.

**Phase 2: Space Rat Tracking**

1. **Initialize rat knowledge base (Rat KB):** Create an initial rat knowledge by initializing the probability of the rat’s position to be equal across all open cells.
2. **Use Space rat detector:** 
   1. Use the space rat detector to listen for a ping
   2. Update the rat knowledge base based on ping or no ping as mentioned above.

**3.Move towards the rat:**

1. Move towards the cell with the highest probability in the rat knowledge base.
2. Update the bot’s position and increment the movement count.

**4.**Continue 2 and 3 in alternating time steps until the rat is caught.

**Design and Evaluation of My Test Bots**

First, for our choice of our optimized bot I developed a preliminary test bot, Test Bot 1, from our Baseline Bot where instead of trying to hear a ping and travel in other time steps I decided to listen for a ping for some threshold, unless α = 0 since the probability is zero regardless of distance between the bot and the rat. For our Test Bot 1, I have defined the threshold as equal to some number of pings ρ. For the sake of experiment, I have chosen 200 ship configurations and placed our bot and rat randomly in different empty cells. For various values of α spread evenly between 0 and 0.5, I simulated both the Baseline Bot and Test Bot 1. Figure below illustrates the performance of both Test Bot 1 and Baseline Bot for various values of α when ρ = 100. Performance in Baseline Bot worsens in as α rises, reaching a peak of poor performance at about 0.2, with some recovery as α reaches 0.5. Here, we can observe that our Test Bot 1 is on average taking fewer time steps on average over various ship layouts to locate its own position and catch the rat. Test Bot 1 demonstrates greater efficacy in reducing the amount of total steps taken over time relative to Baseline Bot, particularly at the mid to high alpha values. Test Bot 1 is always better than the Baseline Bot's performance, particularly considering the low to mid-level alpha values (0.2–0.5). Such a finding validates enhanced efficiency and resilience over the Baseline Bot.

A graph with blue and orange lines

AI-generated content may be incorrect.

Since both Baseline Bot and Test Bot 1 follow the same strategy in Phase 1, i.e., to first identify their respective locations by using alternate time steps to determine the state of the neighboring eight cells in terms of blockage, the total number of time steps required for correct location identification remains the same. So, we consider the average moves needed to trap the rat and the frequency of usage of the Space Rat Detector. By considering the average motions required to catch the rat and the average usage of space rat detector from Figure, we see that although the total counts of time steps required have decreased in our Test Bot 1 for larger values of α (0.4-1.5); however, the quantity of movements needed has risen substantially, even in instances where the frequency of space rat detection is considerably reduced.

We can conclude from here that for higher value of α (0.5 and it can be more if value increase) Baseline Bot when the probability of hearing a ping is very low, the bot makes fewer moves when it keeps listening and moving compared to listening for ρ time steps and then moving. However, since on average the Test Bot 1 is requiring less time steps we can consider that the employed time steps to listen for the pings might not be an overhead for larger α values. However, Test Bot 1 has much steadier and lower movement counts for most of the values of α, especially from 0.06 up to 0.3. Thus, Test Bot 1 has a clear advantage in reducing movements, revealing better strategies for navigation and localization. In the next experimentation, I varied ρ values from 0 to 200. Figure illustrates the average total time steps taken by the bots to localize themselves and capture the rat. Some interesting observations can be observed in this respect. For ρ = 150 the Test Bot seems to perform better than most of the Test Bots on average for α = 0.01 to 0.02. For α = 0.03 to 0.5 Test Bot seems to perform better for ρ = 200. But as α rises as the probability falls exponentially listening for the ping seems to have no impact and thus adds the time step unnecessarily. Consequently, it is observed that after the value of α exceeds 0.4, the best performance is by the Test Bot with ρ = 0.

A graph with a line and a line graph

AI-generated content may be incorrect.

Based on the previous evaluations on Test Bot 1 I have defined the threshold ρ as a function of α to optimize the bot’s performance using equation 3.

The function reflects the relationship between the space rat detector’s effectiveness basedon α, and the optimal number of pings (ρ) required for efficient detection and movement. When α = 0 or α > 0.7, the detector provides either no differentiation (ping always present for α = 0) or becomes almost ineffective (ping rarely heard when α > 0.7), so ρ = 0 ensures no reliance on the space rat detector. For 0 < α < 0.3, the detector is moderately reliable, and ρ = 150 helps the bot to update the rat knowledge base for a while and then move based on that. For 0.3 ≤ α ≤ 0.7, the detector is most effective, allowing for a slightly higher threshold ρ = 200 to utilize the detector before moving.

I tested the bot using two exclusive samples of 20 ships and the performance is shown in Figure. In bot cases, we can see that out bot outperforms the baseline bot for α greater than 0.1. Hence, I performed a more in-depth analysis on our bot for α in between 0 and 0.1 and see if we can find a way to improve our bot’s performance in that area, especially by testing it on Sample 2.

Now, let’s do a more in-depth analysis of the bot’s performance for α between 0 and 0.1. Given for α = 0 my bot is performing well with ρ=0, α = 0.001 to 0.1. In Figure ?? we can notice that our bot is overall performing well except for a few cases.

To solve the issue, I came up with a threshold for which the bot will try to listen for a ping. The threshold depends on the mean probability of the space rat detector and takes into consideration the value of α. The bot decides to listen to the ping detector until the maximum probability of the rat knowledge base surpasses the threshold. Hence, the bot moves after if notices that there’s a cell that has a high probability of having the rat. The threshold is given as follows:

Here, µ is the mean probability of all possible locations in the rat knowledge base.

η is an adjustment parameter that influences how the value of α affects the threshold. I initialized η to be 0. 5. In Figure we can notice that the bot’s performance got better by several degrees, especially alpha has a much lower value. However, it worsens by a smaller degree in some cases.

I tried taking a higher threshold value by taking the value of η from 0.4 and 0.9 to see if the performance gets better or not. In Figure we can see that lowering the threshold increases the number of steps required for lower value of α. However, the average number of steps decreased for α = 0.01, but the amount is negligible.

Hence, I decided to check how the bot performs for the Sample 1 ships for η = 0.5 and from Figure we can see that our bot outperformed the baseline bot for all values of α, except 0.1. However, the difference is quite negligible and hence to finalize I used η = 0.5.

**Algorithm: My Bot**

Here’s the finalized algorithm of the bot that I have designed. Similar to the baseline bot my bot works in two phases:

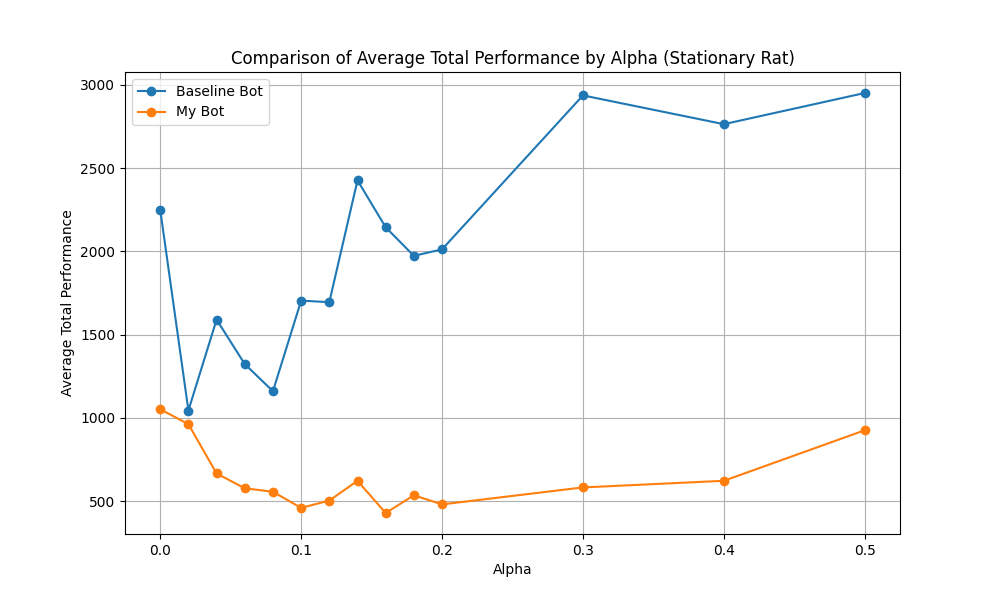
• Phase 1: Identify where the bot is.

• Phase 2: Track the space rat

**Phase 1: Bot Localization**

1. **Initialize Knowledge Base (KB):** Create an initial bot knowledge base with all open cells as possible locations for the bot.
2. **Sense Blocked Neighbors**: Sense how many of the eight neighboring cells are currently blocked.
3. **Rule Out Locations**: Update the bot knowledge base by ruling out locations that do not match the sensed value.
4. **Find Common Direction:** Identify the direction that is most commonly open among the remaining possible locations in the knowledge base. Select the direction that has the highest count of opening cells. However, if there are more than one such direction, the bot selects one at random.
5. **Attempt to Move:** Attempt to move to the most common direction.
6. **Update Knowledge Base Based on Attempted Move:** If the move is successful, rule out all the locations where the move was blocked. If unsuccessful, rule out all the locations where the move was not blocked.

The bot keeps repeating step 2, 3, 4, 5, and 6 until only one possible location remains in knowledge base which is the bot location.



**Phase 2: Space Rat Tracking**

**1. Initialize Rat Knowledge Base (Rat KB):** Create an initial rat knowledge by initializing the probability of the rat’s position to be equal across all open cells.

**2. Calculate ρ:** Calculate a threshold value ρ based on the value of α

**3. Check for α:**

(a) If α is between 0 and 0.04, then:

1. Calculate the threshold value using the equation:

where µ is the mean probability of all cells, and α is the sensitivity parameter of

the detector. η is an adjustment parameter that influences how the value of α

affects the threshold. Here, η = 0.5.

ii. The bot hen computes the max prob which is the highest probability in the rat knowledge base.

iii. If max prob < threshold:

* + - Use the space rat detector to hear for a ping.
    - Update the rat knowledge base based on ping or no ping as mentioned above.
    - Continue using the space rat detector until max prob is greater than the threshold value

iv. Else:

* + - Move towards the cell with the highest probability in the rat knowledge base.
    - Update the bot’s position and increment the movement count.

v. Continue (a) ii, (a) iii, and (a) iv until the rat is caught.

A graph with a line and a line graph

AI-generated content may be incorrect.

(b) If α is 0 or greater than 0.04:

1. If ping count < ρ:
   * + Use the space rat detector to hear for a ping.
     + Update the rat knowledge base based on ping or no ping as mentioned above.
     + Continue using the space rat detector until max prob is greater than the threshold value.
2. Else:
   * + Move towards the cell with the highest probability in the rat knowledge base.
     + Update the bot’s position and increment the movement count.
3. iii. Continue (b) ii until the rat is caught.

Heuristic Considerations - The threshold is dynamically adjusted based on the sensitivity parameter α. A lower α implies a higher sensitivity to distant pings, resulting in a higher threshold. - The bot balances between continuing to gather information (pinging) and acting (moving towards the rat). - The bot uses a two-phase approach: first, to determine its location and then to track and catch the space rat. By dynamically adjusting its behavior based on α, ρ and the calculated threshold, the bot effectively balances exploration and exploitation.

**Moving Space Rat**

**Update space rat knowledge base when the rat is moving:**

The rat knowledge base is a dictionary of open cells as keys and the values as probabilities of having a rat in that open cell. The knowledge base is initialized and updated in the following steps:

**Step 1:** Initialization At first, the rat knowledge base is initialized with all open cells having an equal probability of containing the rat. The probability distribution is initialized as follows.

Where Nopencells is the total number of the open cells.

**Step 2:** Prediction (Rat Movement) Before updating the knowledge base based on observations, account for the rat’s movement. The rat is assumed to move to one of its neighboring cells with equal probability. For all Lj ∈ Lopen cells, the probability is updated as follows:

Where is the transition probability from , defined as:

This step redistributes the probabilities in the knowledge base to reflect the rat’s possible movement to it’s open neighboring cell.

**Step 3: Update Based on Observation**

**(a) Check Current Cell:** If the bot does not find the rat at its current location Lbot, the probability of the rat being in the bot’s cell is set to 0. The probabilities in the knowledge base are updated as follows:

For all other location :

**(b) Ping-Based Update:** The bot uses the space rat detector to listen for a ping. Depending on whether a ping is heard (ping = 1) or not (ping = 0), the bot updates it’s belief using Bayesian inference:

Here is the breakdown of how the update is done: If the bot hears a ping:

A graph with blue and orange lines

AI-generated content may be incorrect.

If the bot does not hear a ping:

The likelihood probabilities are computed as:

The normalization factors are:

**Step 4:** Move Toward the Cell with the Highest Probability The bot moves toward the cell with the highest probability in the knowledge base:

Iteration The bot repeats Step 2, 3, and 4 until the rat is caught. If the bot is not able to catch the bot under certain number of moves, in our case 20000 we are terminating the bot by returning” Rat Not Found”.

**Baseline vs My Bot Performance**

In Figure we can notice that the baseline bot is performing well for lower values of α. However, the baseline bot was unable to catch the rat under 20000 steps in two cases that are shown in Figure.

A graph with blue and orange lines

AI-generated content may be incorrect.

However, the bot I built in the previous section when the rat was stationary is under performing and is getting stuck, while not being able to catch the rat under 18,000 steps. I have tried several methods but have found my bot not being able to catch the bot. My analysis on this is that, as my bot tries to listen for ping for several time steps based on the value of ρ, as the rat keeps changing it’s position trying to hear for ping for 150 or 200 steps at a time is not helpful. Hence, I tried changing the bot design by making the bot hear for ping by alternating listening and moving phases. To represent the alternating behavior of the bot, we define a step-based function that determines the bot’s behavior (listening or moving) at any given step. First, we define the cycle length, C, as the sum of the listening and moving phases:

Where:

: Number of steps in the listening phase.

: Number of the steps in the moving phase.

The baseline bot uses a similar method but with: = 1 and .

During my first test case for rat moving, I tried = 10 and

The bot decides when to move and when to listen by using a function φ(t) that indicates the current phase based on the step t: