**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.