AI PROJECT 2

Name of students:

Chakradhar Reddy Punur (crp190),

Mohammed Najeebuddin Quadri (mq166).

Rutgers University-New Brunswick.
Introduction to AI (CS520).

Table of contents

1. Explicit Formula for Probability Update After a Move Command:

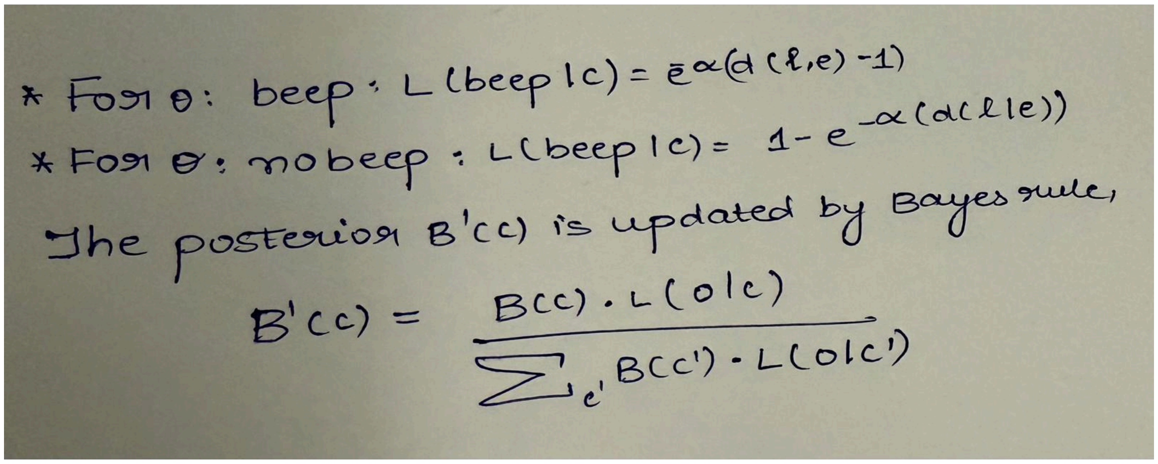
- First, let us assume B(c) as the prior probability that our Roomba is in cell c. Where c is an open cell in the ship.
- Given a moving command like UP, DOWN, LEFT, RIGHT, such that the Roomba attempts to move deterministically from its current cell c to the next cell c' in the given direction m'. If the cell c' is open, then (ship[c'] = 0); otherwise, it stays in c. Defining the function next (c, m) as the resulting cell after this attempt.
- The posterior belief *B'(c)* after the move command completely depends on the collision course. Which means the Roomba enters the locator bot cell, which is represented by *L*.
    - Case 1: In this case, we will be discussing the collision observed where the Roomba enters *L* and we immediately tag it by terminating with certainty, which can be represented as: B'(l)=1 and B'(c)=0 and for all c!=1. This would only occur when the true Roomba is in the position and moves to *l*.
    - Case 2: No Collision observed: In this case, first we compute the temporary belief *T'(c)=∑c:next(c, m)=c' B(c),* aggregating probability over all priors that map to each possible next cell. Since there is no collision, we will remove the probability mass on *L* and renormalize the following formula:

$$B'(c) = \begin{cases} 0 & \text{if } c = 1 \\ \dfrac{T(c)}{1 - T(c)} & \text{if } c \neq 1 \text{ and } T(L) < 1. \end{cases}$$

- From the above image, if T(L)=1, this would imply a certain collision. But in practice, we terminate before updating.

2. Explicit Formula for Probability Update After Running the Detector:

- The detector outputs the observation *O,* where *O* is either beep or no beep. The likelihood *L(o|c)=P(o|roomba in c)* is given by the project's model, using shortest-path distance d(l,c) between locator cell l and c:



* For o: beep : $L(beep|c) = e^{-\alpha(d(\ell,c)-1)}$

* For o: nobeep : $L(beep|c) = 1 - e^{-\alpha(d(\ell|c))}$

The posterior $B'(c)$ is updated by Bayes rule,

$$B'(c) = \frac{B(c)\cdot L(o|c)}{\sum_{c'} B(c')\cdot L(o|c')}$$

- This is the standard normalizing Bayesian update, with no change if the denominator is zero.

3. Design Choices of the Strategies:

Baseline Strategy 1:

The baseline strategy 1 employs a non-probabilistic approach to localize the Roomba by maintaining a belief set, which represents all possible open cells where the Roomba could be, assuming uniform initial likelihood without assigning explicit probabilities.

It starts by initializing the belief set with every open cell in the ship, reflecting the unknown starting position while excluding walls. After each movement command (UP, DOWN, LEFT, RIGHT), the strategy updates the set by simulating the move for every current possible location: if the target cell is open, the position shifts; otherwise, it remains unchanged. This deterministic update aggregates all resulting positions into a new belief set, effectively narrowing possibilities based on the ship's physical constraints. Precomputed shortest-path distances between open cells provide map knowledge for evaluation.

To find the optimal sequence, the strategy uses a Manhattan-span heuristic to estimate belief spread—the sum of the bounding box's width and height serves as an admissible lower bound on remaining moves. It then applies A* search in belief space, where nodes are belief sets, edges are movements, g is the moves taken, h is the heuristic, and the goal is a singleton set. This guarantees the minimum moves to localize without sensors. The output includes the command sequence, which includes the sequence of actions that the strategy issued to localize the bot. Using this, the total number of moves can be calculated.

Baseline Strategy 2:

This baseline strategy would implement the Full Bayesian Localization. The belief is first initialized uniformly over all open cells, and at each timestep, the algorithm alternates between movement and sensing. This movement is modeled for both the hidden true Roomba and to belief update by ensuring the consistent prediction.

The Detector provides noisy distance-based beep readings, which are operated by the Bayesian corrections. A greedy policy chooses actions that allow our Roomba to move. The process repeats until the belief becomes highly concentrated in a single cell. This concept would allow the baseline strategy to form a probabilistic baseline for evaluating more advanced localization methods.

Smart Strategy:

We developed a smart strategy to reduce the total number of senses and actions, which uses two approaches:

- We calculate the entropy of the belief distribution, which measures the uncertainty about the Roomba's location. This is an important metric to consider when deciding between sensing and moving the Roomba.
- Movement and sensing are chosen based on whether the uncertainty is high or low.

a) If high entropy, then sensing:

In each step, we compute the entropy H(p).

If H(p) > threshold, then the belief is spread apart, and moving the bot by the locator is not useful. So, in this case, we use the detector to sense until the belief becomes concentrated.

b) If low entropy, then movement:

If the belief distribution is peaked, then we can issue the command to move. If it is moderately peaked, then we move the assumed Roomba to the most likely cell.

If the belief is somewhat spread, then we move the Roomba to the centroid of the distribution.

We stop when the max belief is greater than 0.99 or when the Roomba collides with the locator.

Theoretical Backing and Justification:

This strategy is based on Partially Observable Markov Decision Processes (POMDP). This is a way for an agent to make decisions in uncertain situations.

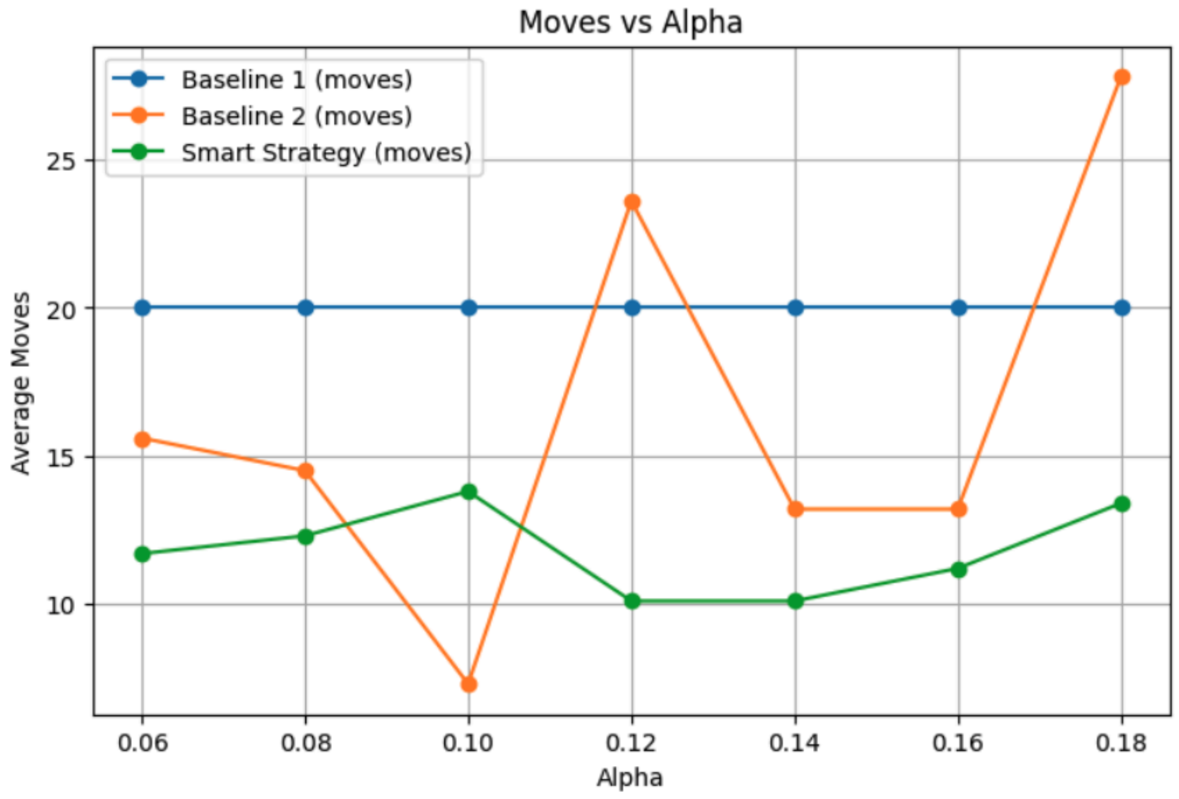
The agent chooses actions like sensing or moving that give new information to reduce entropy. High entropy means there are a lot of possible spots, and low entropy means you're pretty sure.

When alpha is low, it's a noisy detector, so sensing alone is not very useful. Telling the Roomba to move instead shortens the distance between the Roomba and the locator. This, in turn, makes future sensor readings more useful.

When deciding when to move, this strategy uses the centroid of the belief or the average spot in a cluster of probabilities.

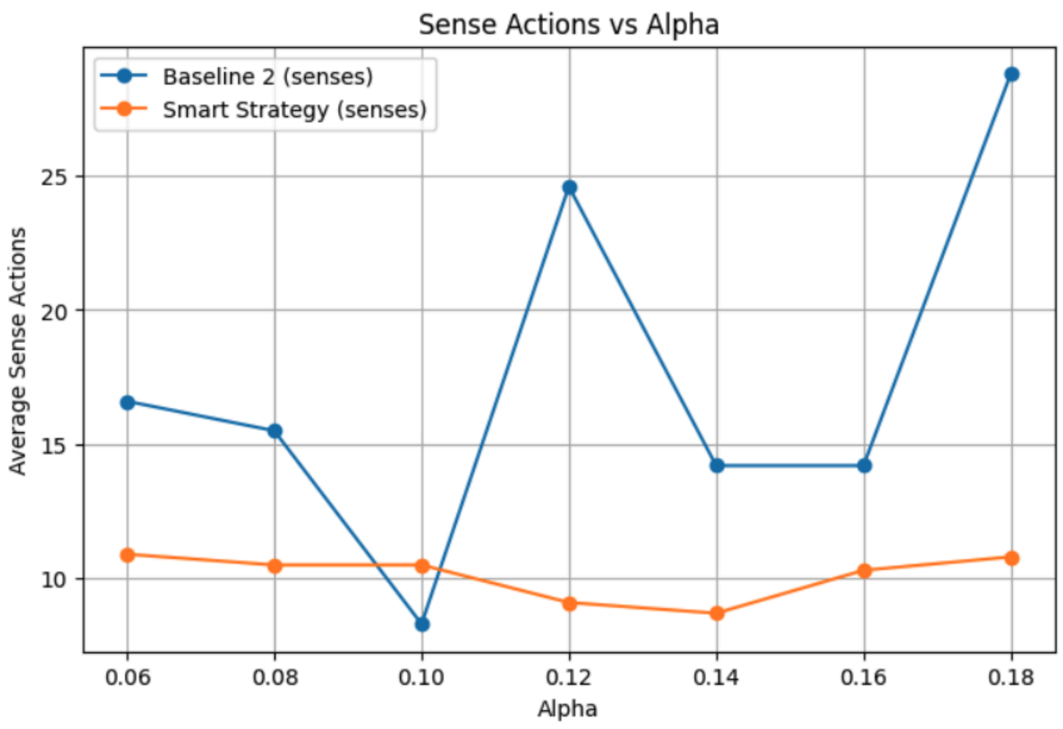
Overall, this strategy senses when confused but moves when the confusion reduces and chooses action with the highest utility or lowest entropy.

4. Analysis:

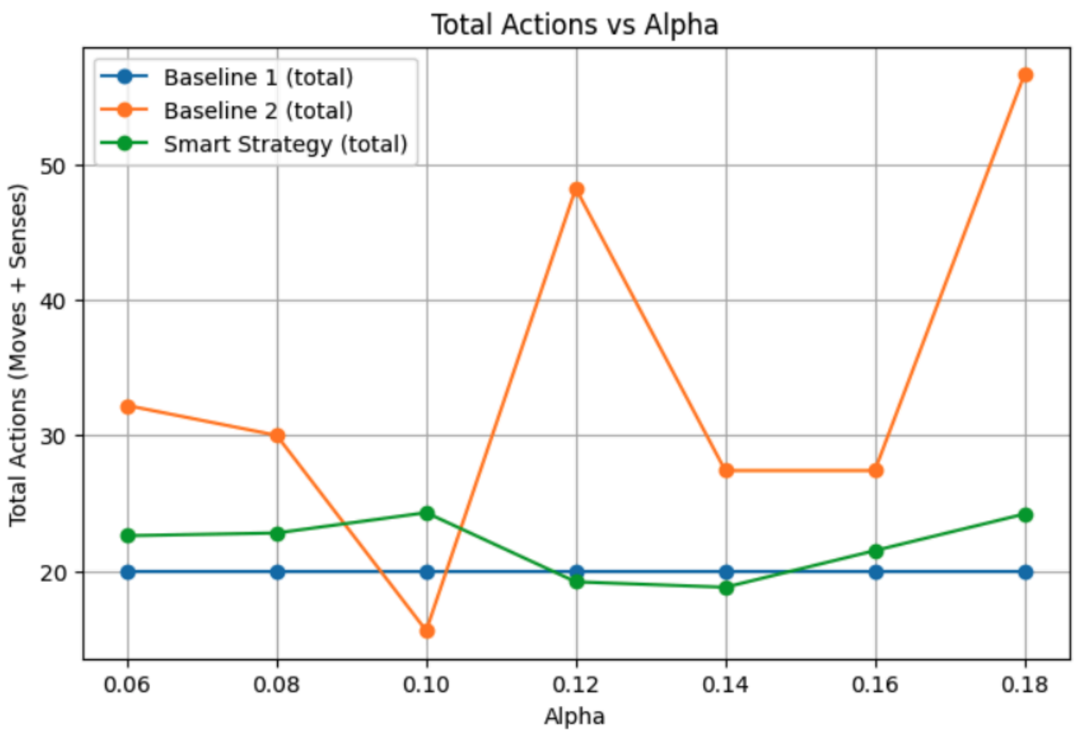


This graph shows the number of moves for all three strategies. It can be inferred that the baseline strategy 1 is constant for all alpha. Baseline strategy 2 gives fewer moves compared to baseline strategy 1, except when alpha is 0.12.

The smart strategy gives the fewest moves for all alpha values except when alpha is 0.1, where the baseline strategy 2 gave the lowest moves.



The above graph shows the sense moves for the baseline strategy 2 and the smart strategy. The smart strategy has fewer moves for all alpha values except when alpha is 0.1



This graph shows the total number of moves, which shows that the smart strategy and the baseline strategy 1 have approximately the same number of total moves, which is less compared to baseline strategy 2. This shows that my smart strategy is smooth and lowest, using fewer steps by being choosy about actions. The separate graphs for moves and senses show similar trends with smart winning by sensing smarter, not more.

5. Fourth Strategy:

- To leverage the locator movement, we have derived a fourth strategy where we treat the problem as a POMDP with some actions. As these actions include SENSE, in which the roomba moves up, down, right, or left, or the locator moves (l-up, l-down, l-right, l-left). This strategy uses the 1-step lookahead to select the action maximizing expected utility.

- The formula can be represented as follows:

$$\text{Score}(\alpha) = \underset{\substack{\downarrow \\ \text{info Gain}}}{\Delta H} + \underbrace{w_d \cdot \Delta D}_{\text{Dist Reduction}} + \underbrace{w_t\, P(\text{term})}_{\text{Termination.}}$$

  Where d/dx (H)= Current entropy minus expected entropy after action.
  d/dx (D)= Expected graph distance before minus after (weight by wd=2α).
  P(Term): expected max belief prob after action.

- When to act:
  - o SENSE: High uncertainty.
  - o Locator move: To probe the high prob cell or to the belief centroid.
  - o Roomba move: to pull the belief mass towards the current locator.
  - o Belief Update move: If we move to new_loc, remove mass from new_loc and renormalize.
  - o Implementation: Precompute graph distances and lookahead uses deepcopy for simulations

- The Experiments were run on a 10x10 grid in order to compare 2 strategies: Smart and Fourth. We measured moves, senses, and total actions across different noise levels (α).
- Main Findings:
  - Smart Strategy:
    - Uses ~20 ~ 25 total actions.
    - Peaks at ~25 actions around α=0.12. Because it oversenses when the noise is moderate.
  - Fourth Strategy:
    - Starts around ~5 actions at low α.
    - Drops to ~3 actions as α increases.
  - Breakdowns:
    - Moves:
      - Smart: ~10 to ~12
      - Fourth: ~ 3 to ~5.
  - Total actions:
      - Fourth is 70 to 85 % more efficient than Smart.
- Why does it reduce the actions:
  - The locator movement transforms the problem from a single-agent POMD to a multi-agent one, expanding the action space to include repositioning for better observation geometry.
  - Noisy Detector Challenge: In this the P(beep_e^{α(d-1)}) decays slowly for low α and large d, yielding low information gain per sense. The locator movement minimizes expected d. This boosted the future sense utility.
  - Probing for Termination: The locator moves to a cell with prob_at>0, offer = prob_at. This short-circuits long sense chains.
  - Belief Renormalization: Post-move updates would remove impossible mass from concentrating beliefs faster than sensing alone.
  - Edge cases: For the disconnected belief or far starts, the locator closes gaps. Without the roomba moves risk the spreading belief further.
- Experimental Setup:
  - Smart Strategy:
    - The locator stays fixed at the center.

        - Only the Roomba can move.
    - Fourth Strategy:
        - Both the Roomba and the locator can move.
        - At each step, the algorithm would choose an action:
            - Sense
            - Move Roomba (UP, DOWN, LEFT, RIGHT).
            - Move Locator (Up, Down, Left, Right)
            - For each noise level α belongs to [0.06, 0.18], so we averaged the total action (moves+senses) across Roomba starting positions on the same.
            - Plot meaning:
                - X-axis: noise level α
                - Y-axis: average total actions.
                - Blue: Fourth strategy.
                - Orange: Smart Strategy.
- What the results show:
    - For the low **α ((0.06,-0.14)),** both strategies depend only on the **α values.**
    - For the high **α (0.16-0.14)** are the smart strategy remains stable on 21-24 actions. But the 4th strategy collapses when 2 actions are required.
- Why does this happen:
    - When **α is too small.** The detector is weak. Moving the locator doesn't provide enough new information to justify the action cost, so mobility offers little benefit.
    - When **α is moderate.** Moderate α (0.10-0.14): The detector is strong enough that the target can be found efficiently with just Roomba moves and sensing. Moving the locator is often an unnecessary action.
    - When **α is large.** High α (0.16-0.18): The detector is extremely sharp. The strategy can quickly move the locator directly to the most probable area, where a single, highly informative sense is then enough to pinpoint the target, leading to very few total actions.

Collaboration Note

This work was done collaboratively by both Chakradhar Reddy Punur and Mohammed Najeebuddin Quadri. We jointly developed the code, performed the analysis, ran experiments, and wrote the report.