Caden Brinkman

Intro to AI

Dec 2nd 2024

**1.1 Task 1**

MDP Project

Value Iteration Rewards per square:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Utility | Change Utility? | Reason |
| Goal | 20 | No | Reward a lot for being close to the goal (relative to the empty reward) |
| Death | -1000 (can’t be -inf) | No | Punish hard for going to a death square, while still allowing high nearby utilities (even if very unlikely) (can’t use -inf because it eliminates going a direction) |
| Empty | -0.2 | Yes | gives us a reference point while also punishing loops and long paths |

**1.2.1 Approach 1: Max Expected Utility**

Using the max expected Utility works very well once we gather enough information about where we are located. Otherwise, the algorithm has a hard time balancing between two actions. For example, if the board is large, actions that guarantee navigation away from danger will be ignored because the probability that the AI is in a nearby location is too low. Also with big boards, the algorithm has a hard time knowing where it is and thus stays still for long periods. Overall, if we don’t know where we are, the algorithm has a chance of getting stuck in the same position which makes the AI take a long time to find the goal. This fact can also be seen in the video where an inaccurate sensor requires the AI to keep still until probabilities are high, while an inaccurate movement simply requires the AI to retrace it’s steps. The combination of the two thus requires the AI to stop to asses it's surroundings every time it makes a movement mistake. Compared to a person’s operation of the robot, the AI takes a lot of time exploring by staying still whereas a person will take risks and move in one direction until reasonably knowing where they are. Plus, the person would try and eliminate possible locations by looking for differences between paths.

Trying it out for different values of pm and ps (in order of video) (unknown initially):

|  |  |  |  |
| --- | --- | --- | --- |
| pm | ps | World | Moves |
| 1.0 | 1.0 | Mundo\_maze | 26 |
| 1.0 | 0.8 | Mundo\_maze | 27 |
| 0.8 | 1.0 | Mundo\_maze | 33 |
| 0.8 | 0.8 | Mundo maze | 56 |
| 1.0 | 1.0 | Mundo maze2 | Inf… (got stuck in place) |
| 0.9 | 0.9 | Mundo maze2 | 85 moves (died) |
| 0.9 | 0.9 | Mundo maze2 | 365 moves (won) |

1.2.2 Approach 2: Finding walls then Max Utility:

For my custom approach I made my algorithm move upwards and to the left at random intervals until my algorithm had a high enough probability of where it was located. Afterwards, it would use the max expected utility algorithm again, but because of the location prediction the AI would start moving towards the goal faster. This entirely depends upon the layout of the maze however. If there are any death squares near the top left of where the AI spawns initially, the algorithm will most likely not find the goal. This is similar to how I controlled the robot initially, however I would move in an initial direction that would minimize the probability of moving into the death squares every time instead of just moving up and left.

The probability bound used was 0.2 (if any square had a location probability more than this we use max utility)

|  |  |  |  |
| --- | --- | --- | --- |
| pm | ps | World | Moves |
| 1.0 | 1.0 | Mundo\_maze | 31 |
| 1.0 | 0.8 | Mundo\_maze | 30 |
| 0.8 | 1.0 | Mundo\_maze | 40 |
| 0.8 | 0.8 | Mundo maze | 44 |
| 1.0 | 1.0 | Mundo maze2 | 69 |
| 0.9 | 0.9 | Mundo maze2 | 142 (death) |
| 0.9 | 0.9 | Mundo maze2 | 122(death) |
| 0.9 | 0.9 | Mundo maze2 | 163(death) (stuck at possible death going downwards) |