Matt Christensen

Carter Madsen

Sihnyoung Park

The Value Iteration extension of the lab was really interesting! We followed the prescribed formulas really closely, and those did us a lot of good. We did run into some hiccups along the way- here are some of the biggest examples:

We had to rehash the Value Iteration algorithm a number of times. The first time we ran it, it did technically build out a trail of increasing utilities from the robot to the goal, but had some issues with creating zones of extreme negative utility while it did so. In this design, the algorithm would sum its utility with its highest-utility neighbor's future reward- which, as all of our floor spaces started out weighted at -1, led to floor spaces to drop in utility during each run of Value Iteration. We were able to get this fixed up, and got more normal utilities after that!

We also ran into trouble when we realized that we weren't adding up each floor tile's immediate utility with its future reward, but rather treating both values as one & the same. While it looked like a good enough idea on paper, we realized our mistake when floor tiles continuously summed up the rewards from their neighbors, leading to perfectly average floor tiles with reward values many times that of the goal itself! We corrected our algorithm by splitting our calculations across 2 arrays (one to hold immediate rewards & one to hold future rewards), and then things started to work as normal.

We had a debate while developing the algorithm about how routefinding should be done. In full transparency, I (Carter) hadn't properly accounted for the nondeterminism of the robot in my thinking, and as such didn't factor the possibility that the robot may end up moving in an incorrect direction or that it could get a noisy sensor reading into my thinking. Had we gone ahead with my plan, the robot would have simply followed a breadcrumb trail towards the goal

without taking the effort to avoid potential pitfalls. It would have been much simpler, but it would have been critically flawed- the robot could have skirted the edge of stairwells, just one bad motor movement away from its demise.

Interestingly enough, the robot is more careful than humans might tend to be right off the bat. Because we re-used our transition model function inside of our Value Iteration function & took into account the probability of making a false move, stairwells now have a “buffer zone” around them and, when given the choice, a robot will stay away from the squares right next to stairwells due to the possibility of erroneously falling in if a bad motor movement is made!

The robot performs its job well. It’s interesting to note how much sensor noise & motor noise has an impact on its performance, though- performance drops off rather quickly once we don’t guarantee the robot knowledge about its position. If we keep the sensor inputs high enough, though, it does a good job finding its way back home!

Video links:

https://github.com/mrchristensen/BayesFilter#automatic-value-iteration-algorithm-decides-where-to-go