Robot Learning 2023

02288063

Coursework 2

Description Of Your Implementation

High-level summary

Training

Training is split into two phases, the exploration phase and the safety-net phase. The first phase takes about the 5/6 of the training time and the second phase the remaining 1/6.

The exploration phase is based on the idea of the knowledge-based intrinsic rewards and the main task is to train the model as well as possible. It encourages the robot to explore areas with high uncertainty. This exploration technique has a significant side benefit, when the robot tries to get into the obstacle area, it slows down, lowers uncertainty on the edge areas around the obstacle and therefore doesn't enter the area.

During the safety-net phase the robot is reset to its initial position and uses a planning with gradients technique to go straight to the goal state. The safety-net phase serves two purposes. First, it makes sure that the robot familiarises with the area towards the goal state in case it didn't explore it already. Second, in case the robot doesn't reach the goal state, it stores the remaining distance. This gives a vague information about the size and the position of the obstacle which will be later used during testing time.

Testing

Testing is split into two phases, the planning with sampling phase and the planning with gradients phase.

The planning with sampling phase uses a cross-entropy method with open-loop planning (since closed-loop is too heavy computationally) with adaptable planning horizon. Therefore, the performance of this phase relies heavily on the model training. The reward function combines distance from the goal state, from the initial state and uncertainty. Also, instead of following the cross-entropy path, the robot follows a rather straight path to the final state of path of cross-entropy. As soon as the robot reaches the final state of the path, the next phase begins. The planning with gradients phase forces the robot to move towards the target similarly to the second phase of training.

Interesting findings and ideas

Ideas that are included in the final solution

- When moving towards a state with higher uncertainty or that is closer to a state, taking one step towards every possible angle is very unstable. Therefore I implemented a method that takes 10 steps towards 12 possible angles and choses greedily.
- In most cases, a single cross-entropy path is sufficient to navigate around the obstacle but in some cases a larger planning horizon is required.
- Penalising paths with high uncertainty helps avoiding paths inside the obstacle as the area inside the obstacle is not explored during training and thus has high uncertainty.
- Rewarding paths with high distance from the start motivates the robot to plan straighter paths towards a certain direction and less wiggly.
- Following the path of cross-entropy is time consuming, for this task the robot can navigate from the initial state, straight to the final state of the path. This saves valuable time and is almost risk free as the paths proposed by the planning avoid the object.
- By exploiting the info about the size and position of obstacle, the planning horizon can be adjusted. Larger for difficult obstacles and smaller for easier ones.
- Use of custom timers in training and testing that count robot's steps to be able to split training and testing time and switch between strategies.

Ideas that were tested but are not included in the final solution

- Split the cross-entropy path in segments and create checkpoints that robot should reach before reaching the final destination.
- Multiple consecutive cross-entropy methods from the state that the robot stops