Implementation and Evaluation of Monte Carlo Tree Search Methods for Motion Planning

Background

Motion planning for autonomous driving is a complex task. Planning algorithms face a large, continuous state space and have to provide a feasible and safe trajectory in real-time. Reachable sets that contain all states reachable by a vehicle at a specific point in time, can be used to provide formal guarantees especially regarding safety. The reachable set of a controller of the vehicle can be computed by making assumptions about the limits of the control inputs and disturbances by the environment [1]. If the reachable set of the controller does not intersect with the reachable set of other traffic participants, the calculated trajectory is guaranteed to be safe.

However, online computation of the controller and the reachable sets does not meet the real-time constraints of an autonomous vehicle. In order to reduce the online computation time, controllers for time and position invariant motion primitives and their corresponding reachable sets are computed offline and combined online to find a trajectory from a start to a goal state as shown in Fig. 1. However, solving the puzzle of finding the right sequence of motion primitives that optimally connects the start and goal, requires to search through a large combinatorial space. As often a reference trajectory disregarding the reachable sets can be quickly obtained, we aim to use this information as additional guidance for the searching process.

Monte Carlo tree search (MCTS) has shown remarkable performance in game play of high complexity games such as for example in Go [2], but also in motion planning for autonomous driving [3], [4]. It asymmetrically grows a policy tree of best actions, while not requiring an admissible heuristic compared to classic search algorithms like A*.

Figure 1: Discrete online trajectory planning using offline generated motion primitives [1]

Description

This thesis aims to develop an approach on matching a sequence of motion primitives on a given reference trajectory. MCTS should be used to efficiently navigate through the large...
search space spanned by a maneuver automaton and motion primitives. The approach should be implemented in the CommonRoad framework\(^1\) [5], for which a codebase of classic tree searching algorithms (A*, DFS, BFS) exists. As a first step, a plain, extensible MCTS implementation, adapted to the domain-specific requirements should be created. In a later step, different enhancements for MCTS should be applied and compared to improve the runtime of the algorithm.

**Tasks**

- Familiarizing with the literature on MCTS and motion planning using motion primitives
- Develop an extensible architecture for an MCTS that allows to easily integrate different selection policies, default policies and cost functions.
- Implement the architecture based on commonroad-search\(^2\) (used in the exercise of the AI lecture) using Python
- Evaluate and compare the performance of different enhancements introduced for MCTS (see [6] for an overview)

**References**


\(^1\)https://commonroad.in.tum.de/
\(^2\)https://gitlab.lrz.de/tum-cps/commonroad-search