AImotive: Planning and Decision Making assignment

1 Algorithm

1.1 Route Planning

I defined two types of constraints in the search space: 1st type -> Given point (End of the crossing) must be left before a specified time; 2nd type -> Given point (Start of the crossing) must be reached after a defined time. With these two types of constraints we can handle the crossing cars from the right hand side. Examining the possible routes taking into account these constraint types we can see there are three ways of crossing the intersection: Ahead of the first car (1st type constraint), between two cars (1st and 2nd type constraint), After the last car (2nd type constraint).

The motion planning algorithm of my solution is based on two motion primitives. The two primitives are: 1 -> Keeping speed for some time, 2 -> Reach a specified speed with 0 acceleration. The "Ahead" scenario's optimal solution in searched with just two primitives (set speed -> keep speed), since it makes no sense to decrease speed here. (It is always possible, that we do not reach the required speed because we reach the goal, and the keeping speed primitive can have 0 length.) The "Between" and "After" scenarios are built from 4 primitives (set speed->keep speed-> set speed-> keep speed).

The Optimization is done by a simple stochastic method called Chemotaxis. The arguments of our cost function are the parameters of the corresponding motion primitive vectors. (Ahead -> *Maximal Speed*), (Between, After -> *Speed1, Time of keeping Speed, Speed2*). The Optimization finds an approximately optimal solution for every scenario, and returns the best trajectory.

1.2 Prediction:

The prediction part is really simple, can and should be improved further in time. Currently it creates a naïve prediction for the cars on the right side of the ego car, using their actual speed and their distance from the intersection start and end, then extends these time constraints by an uncertainty factor. This way the constrained time in the search space is higher if the car is further (in time) from the intersection. If there is a car on the Opposite lane of the ego car the leaving time estimation of the cars on the right hand side are increased.

E.g.: $T_{LeaveNaive} = \frac{V_{Act}}{S_{CrossingEnd}}$ $\quad T_{Arrive} = T_{LeaveNaive} * e^{T_{LEaveNaive} * C_{uncertanity}} + addTime$

1.3 Control:

The Control part is an interface between the Route Planning, the Prediction and the Gazebo Client. It can be set to call the Route Planning in every x time steps (currently configured to every step). The Control module handles also if there is a car ahead in our lane, by decreasing the speed to the speed of the car ahead if necessary. If there is no route to the goal, the Controller initiates a full brake or keeps speed if the ego car is in the middle of the intersection.

2 Project Structure

The project has 3 parts: The Auxiliary library which contains the Controller, the Route Planner and the Predictor; the Auxiliary_Test, which contains Google Tests for these module, and the client controller (planning_assignment folder), which is basically a modified version of the provided example client. The first two parts are eclipse projects, but can be built with make without the CDT. The Gazebo Client can be created by CMake and build by make. Additional Information can be found on the github repository:

https://github.com/MilanMoro/PlanningSolution

Milán Móró