Path-Planning-Project

The rough idea of the solution is to generate several trajectories, assign a cost value to each trajectory and take the one with minimum cost. The cost of a trajectory is the weighted sum of several cost functions. Cost functions are defined in cost_functions.cpp and are added in main.cpp via PathPlanner::register_cost_function. The next trajectory is computed via PathPlanner::plan. First, we copy 20 points from the previous trajectory and compute the state of the car after following those copied points. Then, this trajectory is expanded by PathPlanner::generate_trajectories. Here, we sample several points around the lane center for each lane (the d-value in Frenet-Coordinates) and iterate for each over several (de)acceleration-values. For each combination of d-value and acceleration-value a, the actual trajectory is computed using PathPlanner::compute_jmt. Here, we estimate the s-value of the car at time T=1.5s (t=0 corresponds to the state after following the copied points from previous path). The (s, d) coordinate is then transformed to the global XY-Coordinate-System (using getXY) and is then transformed to the local car coordinates (x_{end}, y_{end}) system (using compute_local_coords). Using a reference point $(s + 0.5, d)$ we estimate the local yaw γ of the road at the estimated end point. We compute polynomials x(t), y(t) to generate a path in the local coordinate system. For $x(t)$, we use a quadratic polynomial where we set the conditions

$$x(0) = x_0,$$

$$\dot{x}(0) = x_0,$$

$$\dot{x}(T) = \cos(\gamma) * (v_0 + a) .$$

For $y(t)$, we use a cubic polynomial with the conditions

$$y(0) = y_0,$$

$$\dot{y}(0) = y_0,$$

$$y(T) = y_{end}$$

$$\dot{y}(T) = \sin(\gamma) * (v_0 + a).$$

To compute the final trajectory, we evaluate $x(t), y(t)$ at $t = i * 0.02s, i \geq 1, t \leq T = 1.5s$ and transform back to the global coordinate system (using compute_global_coords). The cost of a trajectory is computed via PathPlanner::compute_cost, additionally, we reduce the cost (i.e. multiply by 0.5) of a trajectory if its goal lane is the same as planned before (the idea behind is to only change lanes or abort a planned lane change if there is a good reason, i.e. significant lower cost value). The cost functions which I finally used have the following goals:

- speed_cf: Keep the car close to the target speed
- keep_lane_cf: Keep the car close to the lane center (of current or goal lane)

- min_lane_changes_cf: Minimize the number of lane changes, especially more than one lane change within the current trajectory
- collision_cf: Avoid collisions
- off_road_cf: Keep car on the road, don’t come too close at road boundaries
- exceeds_a_y_cf: Don’t exceed max. acceleration in cars local y-direction
- other_veh_gap_cf: Keep distance to other vehicles in current/goal lane
- exceeds_max_speed_cf: Don’t exceed max. speed

The path planner is able to drive the required 4.32 Miles (see ScreenshotFinal.png) and is also able to smoothly change lanes (see ScreenshotLaneChange.png).