Trajectory Planning on Highways

Planning and Decision Making in Robotics - 16782

Danendra Singh(danendrs), Shivang Baveja(sbaveja), Yu Chi Wang(yuchiw)

December 12, 2018

1 Problem

In recent years, autonomous driving has become a center piece of robotics development and has potential applications in many industries, ranging from cab hailing to logistics. In this project, we examine the problem of trajectory planning on highways for autonomous vehicles. Specifically, given a n-lane road and m moving cars on that road, our problem is concerned with the optimal trajectory that the ego vehicle should take.
This problem is challenging because of following reasons:

- State of the vehicle x, y, θ, v, a has 5 degrees of freedom.
- To avoid dynamic obstacles, the planning has to be done considering time as part of the state.
- The planned trajectory should be kinodynamically feasible for the vehicle considering the limits on maximum curvature and max steering rate.

Our goal is to find a safe and time-efficient trajectory. Also, the planner should be able to decide which lane to change to and when.

2 Solution

Finding an optimal trajectory with dynamic obstacles is a np-hard problem. To solve with reasonable planning time, we have made a few simplifications. Nevertheless, these simplifications in no-way diminish the complex nature of the problem. These simplifications are:

- We are limiting the problem to 3 lanes and the start state of the vehicle is one of these lanes with vehicle completely aligned with the road. So, the planner does not handle re-planning during lane-change.
- In any state, the vehicle can either initiate a lane change(right or left) or keep straight. Also, for each of these actions it can either stay at the same speed, accelerate or decelerate. So in total we have 9 actions.
- For any state the vehicle speed stays within the closed set $\{3m/s, 6m/s, ...21m/s\}$. Also, x-coordinate is discretized into 5m intervals, and y coordinate for any state is either -4m, 0m or 4m.
- The ego-vehicle and other vehicles on the road are assumed to be represented as rectangle. Also, only ego-vehicle changes lane, other vehicles can be spawned in the environment at different distance, lane or speed but they just move straight.
- Planning is done for a distance horizon of 100m and time window of 15 seconds, whichever is reached first. This is done to limit planning time in case the obstacles are really slow.

Following figure shows our formulation of trajectory planning problem on highways:

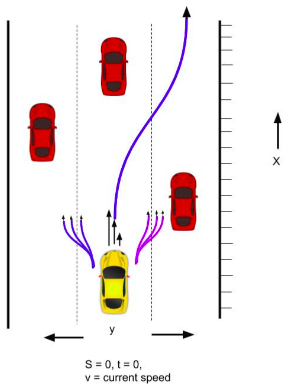


Figure 1: Our formulation of trajectory planning on highways

3 State Space

The native state space is continuous and to be able to play in a time and memory efficient manner, we use a finely discretized version as described above. The complete state space for the planner is the tuple (X, Y, V, T, θ, A), where X and Y are the cartesian coordinates, V is speed, T is time, θ is the heading, and A is acceleration. However, for planning purposes, only (X, Y, T, V) are used as independent variables. The acceleration and heading vary along a trajectory but at any graph node, the heading of the vehicle is 0 degrees i.e. it is completely aligned with the road and the acceleration is 0 m/s^2.

4 Trajectory Generation

For the purpose of generating kino-dynamically feasible trajectories we used the method described in the paper, Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice by McNaughton et al. 2011.

We first find the kinodynamically feasible paths parameterized with curve length s. This is done by solving a boundary value problem between start point($x = 0, y = 0, \theta = 0$) and goal point ($x, y = \pm 4, \theta = 0$). The x-coordinate is increased from 5m to 50m in steps of 5m, to find a path within curvature limits. Once a feasible path is found we apply fixed speed profiles such that the speed either remains same, increases or decreases by 3m/s. If the trajectory violates the limit on rate of change of curvature it is ignored and we continue to increase x. The same process is repeated for right lane change and straight action. At the end we have 9 feasible trajectories at any state.

To solve the boundary value problem, curvature along the path is assumed to vary as a cubic polynomial function of the curve length. For the optimization, we use 3 parameters, curvature at (1/3) arc-length, curvature at (2/3) arc-length, final curvature and the arc-length itself. At the beginning of optimization, curvature parameters are assumed to be 0 and arc-length is set equal to the goal x-coordinate, i.e. we start optimization with a straight line. Also, the cost used for optimization is L2 distance from the desired goal state. During each iteration of optimization, each of the parameters are nudged and response is the final state is computed by projecting the motion model forward from the start state. This gives a gradient of cost w.r.t the specific parameter. We then use gradient descent to update the parameters. The following figure shows the how path changes during optimization. As parameters are updated using gradient descent, we get paths that are converging towards the desired goal.

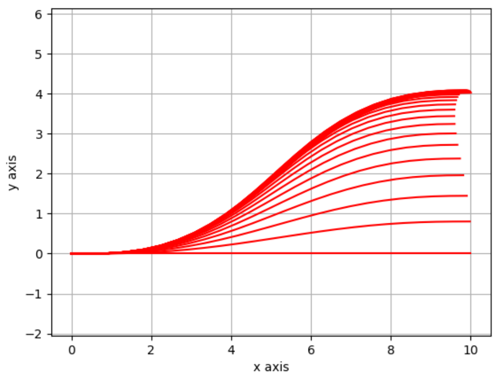


Figure 2: Gradient descent to path optimization

5 Planner, Actions and Heuristic

We are using A^* planner with pre-computed trajectories as successors for each state. The trajectories were computed in python and stored in a json file which is loaded at run time by the planner. The planner was written in c++. For each successor, cost is assigned as a weighted combination of time and closeness to the dynamic obstacles. This is done to keep the trajectories time-efficient and at the same time slightly biased from the obstacles. The heuristic used is computed by calculating the time to reach the goal distance at maximum speed. This heuristic is admissible as it always under-predicts the cost to goal. The goal for A^* is defined as both goal distance and goal time whichever is reached first.

6 Collision checking

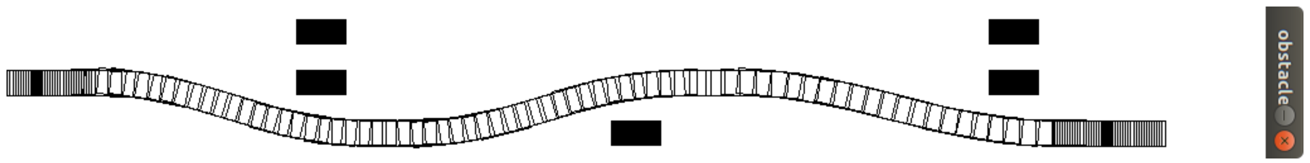
At the beginning of each planning cycle, a distance transform is computed in x, y and time. While computing the distance transform, all the obstacles are first inflated a fixed amount to account for the size of ego-vehicle. The output of distance transform is used to check collision and also to assign a cost of how close a trajectory is to the obstacles.

7 Results

To validate the applicability and correctness of our planner, we tested it on several test cases:

- **No obstacles**: The environment has no other vehicles
- **Fast obstacles**: The environment has other vehicles that travel faster than the ego vehicle
- **Slow middle obstacles**: The environment has other vehicles in the middle lane that travel slower than the ego vehicle
- **Change lane**: The environment forces the ego vehicle to choose a lane
- **Wait and change**: The environment forces the ego vehicle to wait a significant distance before performing a lane change
- **Multiple lane change**: The environment has a lot of static obstacles.

Video links for these results are **here**. For visualization purposes, we have plotted the planned trajectory for the multiple lane change environment below horizontally.



References

[McN+11] Matthew McNaughton et al. “Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice”. In: *Proceedings of the International Conference on Robotics and Automation.* May 2011, pp. 4889–4895.