ESE 615 Final Project Proposal: We Can Learn How to Race

Team 5: Peyman Norouzi, Christopher Hsu, Yuwei Wu

April 2020

1 What is the problem?

Autonomous vehicles are built up of many parts that look to solve the problem of perception, planning, and control. To tackle these parts we will need to define a framework in which to solve these parts. The perception problem is defined as how to identify the environment the vehicle is in as well as the dynamic parts of it. It is necessary to define and identify the parts of the states in which it is safe to drive. Therefore, we need to define a planning stage. The planning stage can be looked as deciding on which path to take to accomplish some goal. Planning is an NP-hard problem and it should not be taken lightly as it has repercussions in safety. Solving the planning strategy is an important portion of this project and will be defined later. Finally, there is the control of the vehicle. Executing the actions decided upon, control will consider the decisions made and act. Initially we will use Pure Pursuit, but if that type of control is not good enough we will consider MPC. Model predictive control is effective in providing not only good, but safe control. Now that we have a skeleton, the goal is to win the race. Therefore, the problem design specifications will cater towards that goal such as the RL reward.

2 Why is it important?

There are a few reasons that indicate the ever more importance of developing autonomy for four-wheel vehicles. The top two of which are safety and efficiency. The future of autonomous vehicles promises a safer driving experience in which human error has been eliminated from the driving experience. Based on a report released by National Highway Traffic Safety Administration (NHTSA) in 2016 [1], roughly %94 of all vehicle crashes were due to human errors such as fatigue, speeding, and drunk and distracted driving. Autonomy could eliminate most of such errors as long as they are designed with safety in mind. Well designed autonomous vehicles are also going to be a lot more efficient for both individuals and the economy as a whole. Since autonomous vehicles don't rely on humans for driving, they could perform their tasks night and day without the inefficiency of humans. One obvious example of such improvement in efficiency is in truck driving. Autonomous trucks will be able to perform their transportation tasks 24/7 while using noticeably less fuel per mile driven. A new study by the University of California San Diego [4] showed that TuSimples' (self-driving trucking company) level 4 autonomous trucks were able to cut fuel consumption by at least %10 which is significant.

3 Why is it hard?

The problem of decision and control is inherently a difficult one. In a continuous space, there is an infinite number of solutions to pick from and therefore very difficult to distinguish between optimal actions. RL approaches seek to use neural networks to learn this nonlinear mapping between states and actions in order to try and search for some solution. RL needs direction in order to be successful and therefore it is important to limit the space in which it needs to search. Defining this space as well as the state is a difficult task as there is an infinite number of ways in which we can define these parts. In the end, we will come up with a framework that limits the scope of RL in order to make it effective to make decisions.

4 What have other tried to do?

Reinforcement learning-based planning is widely applied in autonomous driving, and functional safety is necessary for this procedure. [3] used policy gradient iterations to find the driving strategy while divide the policy function as learnable a non-learnable to ensure the basic safety of the autonomous vehicle. [8] introduced the "Responsibility Sensitive Safety" model to measure safety distance and combined with Reinforcement learning to create a robust system. Considering the uncertainty of the opponent's behaviors in autonomous racing, [9] proposed a population-based self-play method to parametrize the opponent's behaviors and adjust to evaluate different plans. There are two main input methods for the implementation of deep reinforcement learning. We can either input the color image of the environment as what Waymo did in [2] or we can input core features of the vehicles and the maps, such as the speed, vehicle size, lane size, etc[6].

5 What will you do?

To come up with our approach, we reasoned about the driving/racing requirements to come up with a solution that answers our needs effectively. Our solution divides the decision-making process into two high-level categories. The first category is what we call global decision making that allows the car to make a driving decision that would help it to win the race in the least amount of time considering a global understanding of the environment. The other category is the local decision making that smoothly in acts on the global decision-making order speedily and smoothly. You can see the overall algorithm structure in figure 1.

The first step (prerequisite) in our approach is to create a couple (for simplicity let's go with 3 for now) of global and semi optimized paths that go around the track not knowing anything about possible future dynamic or static obstacles (you can think of them as lanes in the track that the car can drive in). For our global decision making, we use Reinforcement learning (RL) based methods to understand the environment (in ego's frame but with global context) and decide which one of the three pre-defined paths is the best at each time step. We got the idea of using a predefined global map with RL from [7] in which Shwartz et al. uses a similar but more comprehensive approach. Implementing RL on autonomous agents is not new. RL has shown to learn the dynamics of the system to produce a controller. The first successes of deep reinforcement learning algorithms came from introducing CNN's on an image-based game. In this project, we will utilize the stochastic nature of Soft Q Networks (SQN). The algorithm learns a policy that maps the input states to actions. We will reinforce the idea of winning the race in our RL reward structure by giving the actor a positive reward for winning each lap in the race. We will also reward the ego car based on its Time-to-Collide (TTC) calculation to its closest object (higher TTC the better). We are considering two approaches when it comes to feeding the environment data to our RL network. The first approach is to feed the actual global map populated with the ego's and advertorial's cars poses through a CNN, similar to the approach taken by Bansal et al. in [2]. The other approach is to feed important information such as ego's and adversarial's pose information (In SI unit) in a vector format as described in [6]. In both approaches, We will utilize information from the global map as an input to our network to teach the agent about its localization as well as the adversaries' true pose. After the decision of picking from the predefined paths is complete, we recreate our path from our current position to the decided path and use the pure pursuit controller described in [3] to control the car through the created path.

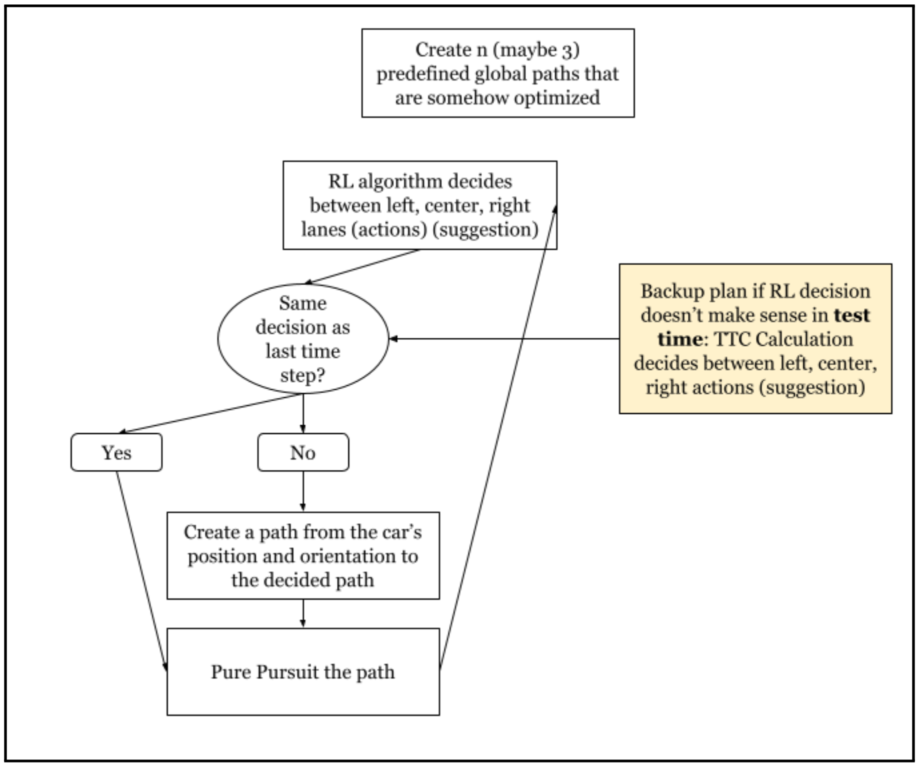
Create n (maybe 3) predefined global paths that are somehow optimized
RL algorithm decides between left, center, right lanes (actions) (suggestion)
Same decision as last time step?
Backup plan if RL decision doesn't make sense in **test time**: TTC Calculation decides between left, center, right actions (suggestion)
Yes
No
Create a path from the car's position and orientation to the decided path
Pure Pursuit the path

Figure 1: Overall Architecture design of our approach

If everything goes well and we have extra time by the end of week 2-3, we will replace the pure pursuit controller with a local MPC controller. Using MPC, we will use the reference trajectory (precomputed trajectories), local lidar information, and linearized car dynamics to come up with the best local trajectory for the car at each time step.

5.1 Soft Q Network

We will use the Soft Q Network version of [5] algorithm to map states to discrete actions or in other words which lane we should switch to. Built on the framework of Double DQN, we use this maximum entropy objective to augment the normal Q and V functions of DQN. The optimal policy π^* can be expressed in terms the optimal Q-function as an energy based model:

$$\pi^*(a|s) \propto \exp\left(\frac{1}{\alpha} Q^*(s,a)\right) \tag{1}$$

The Q-function takes the role of negative energy. To learn the optimal Q-function in this framework, we can formulate the soft Bellman equations for the maximum entropy objective.

$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V(s')] \tag{2}$$

$$V(s) = \text{soft} \max_{a \in A(s)} Q(s,a) = \alpha \log \int_A \exp\left(\frac{1}{\alpha} Q(s,a)\right) da. \tag{3}$$

V(s) is defined using the soft maximum of the Q-functions over actions. Similar to DQN, SQN will learn the parameters of Q function by minimizing the soft Bellman residual which is the difference between the left and right-hand sides of equation (2).

6 Progress Projection

6.1 Week 1:

1. Generate several optimized global paths
2. Pursuit pursuit on path
3. Setup RL training loop, importing environment, plotting

6.2 Week 2:

1. Set up observation state and action space for RL and start training
2. Create a relative distance logic, TTC calculation, for safety and planning
3. Pure Pursuit with obstacle avoidance (TTC calculation). (may add other path planning algorithm)

6.3 Week 3:

1. Combine trained RL policy with pure pursuit for the entire race loop.
2. Implement the local MPC controller to replace pure pursuit (If we have time and everything is going well)
3. Tune/Debug RL with pure pursuit and/or MPC if ready

6.4 Week 4:

1. Tune/Debug RL for race with best controller
2. Tune/Debug our model for the race

References

[1] N. H. T. S. Administration. Traffic safety facts 2016. 2016.

[2] M. Bansal, A. Krizhevsky, and A. S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018.

[3] R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. 1992.

[4] R. GEHM. Self-driving trucks cut fuel consumption by 10%, Dec 2019.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[6] A. Kuefler, J. Morton, T. A. Wheeler, and M. J. Kochenderfer. Imitating driver behavior with generative adversarial networks. *CoRR*, abs/1701.06699, 2017.

[7] S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016.

[8] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *CoRR*, abs/1708.06374, 2017.

[9] A. Sinha, M. O'Kelly, H. Zheng, R. Mangharam, J. Duchi, and R. Tedrake. Formulazero: Distributionally robust online adaptation via offline population synthesis, 2020.