Hello everyone, I am Ce Hao.

In the last semester I presented our project of autonomous racing vehicle, which is an evolving sport quickly developed with autonomous driving techniques. Today, I would like to focus on the topic of trajectory planning and share our latest study.

[np]

The primary target of racing vehicle is wining the competition. Usually, racing cars run in a circle track with multiply competitors. The one who finally achieves least lap time will win the game.

Since the track is much wider than racing car, a driver can select either an inner or outer path to promote velocity and minimize racing time.

Given a fixed track and a fixed car, we should be able to find an optimal trajectory that produces the minimum lap time. The question is how to unearth the ideal trajectory.

[np]

A simple idea is to borrow planning algorithms from autonomous driving. For example, there are many mature planning approaches for driving private cars. However, racing raises different requirements from simple driving.

For private cars, they run in open urban road and highway with moderate speed. So, the comfort and security for the passengers are the top priority. Planning algorithms focus more on obstacle avoidance and path smoothness.

However, autonomous racing cars have no passenger. They run in closed and fixed racing track at extreme speed. The right figure is the world-famous Roborace autonomous racing car. The only goal is to reach higher velocity under the limit of friction.

[np]

A traditional way to planning trajectory is model-based planning approach.

First, we should discretize the track to waypoints, then identify vehicle and track model, and use optimal control to plan path and velocity. In real-time control, we can use MPC to track the reference trajectory.

The biggest advantage of this approach is its stability. As long as we build an accurate model, the optimal results are guaranteed. But there are many hidden states of racing cars, for example engine combustion rate and mass transfer, which we can not easily measure in real-time.

Therefore, another way of learning method raised such as DDPG and Soft actor critic. They use end-to-end learning framework, which observes car states and directly output action at each sample time.

In the following part, I will introduce both methods separately and analyze their performance.

[np]

First, we need to consider the basic states of vehicle on the trajectory. They are position, orientation and velocity at each waypoint. The right figure depicts dynamic states of a vehicle.

Consequently, we can formulate a cost function to plan every state that minimizes lap time. In the equation, Delta t denotes the time to pass two waypoints, determined by car state. By minimizing the cost function, the trajectory is derived.

[np]

Then, we build vehicle dynamics using bicycle model. Here we also consider wind drag force which decelerates cars at extreme speed. We also adopted a hyperbolic tangent function to simulate tire lateral dynamics to accurately predict forces on the tire. In addition, we incorporate vehicle kinetic model in Frenet coordinate to describe relative path to reference.

Here state s is the distance that the vehicle travels along the path. Epsi and ey are relative yaw angle and lateral distance from vehicle center of mass to the reference path.

[np]

Now we formulate the time-optimal trajectory planning algorithm. The cost function Jt minimizes the lap time. It is the integration of lap time from start to end point. But time is not a vehicle state, so we transform the equation from time domain to spatial domain. And the integration limits are the distance of start and end points.

From nonlinear dynamic model, derivative of distance along path is formulated as equation 2. Then we plug equation 2 into 1 and discretizing it by centerline waypoints.

Now, we can minimize equation 3 to derive optimal trajectory. However, since the cost function is nonlinear, we must use computation-intensive nonlinear solver. Besides, nonlinear objective has local minimum, so we might not be able to reach the global optimal in complex situations.

Therefore, another useful method is to divide lap time optimization into two convex objectives to approximate the global optimal.

[np]

Thus, we developed the curvature-optimal trajectory planning algorithm. This method has two steps. First, it maximizes velocity based on the path curvature, and path the velocity to the next step, where a new path minimizing curvature is planned based on the velocity it just received, then pass the curvature to step 1 again.

The idea of minimum curvature path is plain. In a circular motion, when we fix the max normal acceleration, which is friction in our case, smaller curvature can promote higher velocity, so that the lap time is reduced.

This approach iteratively optimizes velocity and curvature of path to approximate time-optimal trajectory. And we can also convexify two objectives to use QP solver to take the place of nonlinear solver.

[np]

First, we can directly maximize the vehicle velocity under max friction. Then, we transform the expression of curvature using approximation. Though the curvature is nonconvex, we have planned velocity in the first step, so here the denominator Vx,k are fixed values during the optimization. Consequently, we can use convex solver to plan them. Besides, we also add equality and inequality constraints of vehicle dynamics and track boundary.

[np]

We implemented an experiment on an L-shaped track. This track is comprised of 4 corners. We compared time-optimal and curvature-optimal trajectory planning and also with Learning MPC developed by Ugo. LMPC uses mixed-integer solver and takes very long time for computation. Time-optimal method achieved least lap time but nonlinear solver also spent much time, while curvature optimal method derived sub-optimal results and it only used 1.6 minutes for computation.

In addition, the minimum curvature trajectory can serve as the initial states for time-optimal planning to accelerate optimization and partially avoid local minimum problem.

But as we can see, all methods took much more time for computation than on-track running, so they must be done offline.

[np]

At the same time, another group started the same topic with us but from another perspective that to use end-to-end RL to control racing vehicles online. Fuchs used RL framework and achieved human-level performance.

He defined the reward function with two term. The first term r prog denotes the centerline progress between two sampling steps. The second term generates negative reward to punish any collision movement. He used soft actor-critic network with 2 layers, each has 256 nodes. The inputs are car states and outputs are two actions. This is a successful end-to-end method, while it has some difficulties.

For instance, the vehicle must collide on the wall to learn how to decelerate, which is dangerous in practice. Besides, the state space is continuous and high-dimensional, so the interaction for learning must be intensive. It took more than 6400 hours for training.

[np]

Finally, we both implemented our approaches in a high- fidelity racing platform Gran Turismo Sport on PlayStation 4. We did experiment using Mazda Demio car on Tokyo Expressway.

We also checked human drivers’ performance in the same scenario as a benchmark. Our approach using curvature-optimal trajectory with MPC achieved best lap time, taking 3.3% advantage over human fastest record. End-to-end learning also achieved human best performance by model-free learning.

It was amazing to outperform human experts while our methods have a problem that they are hard to transfer to other scenarios.

[np]

For model-based planning, we need to identify the vehicle and track model again, and plan trajectory for the whole track offline. For end-to-end RL, it also has intensive interaction for learning a new network. As the vehicle changes, we need to design new reward function to ensure global optimal.

In fact, human drivers do not think in these ways, they neither consider all car states on the whole track, nor only consider one action at one time. They will segment the track into some corners and plan the states at some key points during the corner.

For example, human drivers remember the maximum velocity that they can safely pass a corner. Where to decelerate, and where to accelerate again. This is also a learning process, but human does not consider complex vehicle model.

[np]

Therefore, we borrow this idea from human drivers and proposed segment-based learning approach for trajectory planning.

In preliminary, we segment the track into several corners and straight ways according to track geometry and human path information. So that we can generalize the track and only plan trajectory on each corner. In the right figures, corners are marked by numbers.

However, the trajectory on one corner also influences the initial states of next corner, so we still need to optimize for every corner on the track.

Therefore, we used a hierarchical framework where the high-level plays as a critic to choose the best policies in a library that minimizes cost at each corner. The low-level is an actor that plans trajectory using the policy of key points receive from high level. And it feedbacks cost to high level.

We use A\* graph search to define cost function with heuristic. The cost function has two parts. First part Delta t is the time to pass current corner k, and the second part is the heuristic to pass the next corner. Sk+1 is the length of next corner and Vexit,k is the exit velocity of current corner. By iteratively optimizing trajectory on every corner, we can derive the sub-optimal trajectory for the whole track.

[np]

The next essential problem is planning trajectory on each corner. We can either use vehicle dynamic model or model-free RL, but we also have another option, to plan states on key points.

A basic corner has three key points, Turn in, Apex and Turn out. They mark three stages of passing a corner where path and velocity have dramatic changes. At kth corner and ith keypoint, the policy is comprised of 4 states s, epsi, ey and velocity.

When the enter states and policy are fixed, we can plan the trajectory in Frenet coordinate using cubic spline to smoothly connect every point. In the diagram, the red path must meet the constraints on four points, and the states between them are interpolated by spline.

Then we can project the path from Frenet to Cartesian coordinate. Though the track edge can bound position and orientation of keypoints, we do not know the boundary of velocity. So, here we calculate the curvature of path and plan the velocity bound using friction limits again. Therefore, the velocity at key points and road between them should also be within bound.

[np]

We implemented this approach in simulation of customized tracks with different shapes. It works well in the single corner and it is also efficient in other shapes of track, and we will further compare racing results in real-time control.

The segment-based learning method has obvious advantages. On the one hand, planning key-point policy and spline interpolate are model-free, so that this generalized path can transfer to other tracks. The local trajectory is optimized by learning and the overall trajectory is planned by graph search.

On the other hand, since we only plan for one corner at each time, this enables us to plan trajectory online, and planning key points at corner also reduces the high dimension of car states of end-to-end RL. This approach mimics the decision process of human drivers and further develop online planning algorithm in multi-play competition.

[np]

In summary, this presentation introduced three ideas of trajectory planning for racing vehicles. First, we used model-based method to plan time and curvature-optimal trajectory for the whole track, which requires accurate model and long computation time.

Second, we reviewed the idea of model-free learning. Other researchers used end-to-end RL planning each action according to current states. Both methods achieved super-human performance in the real-time racing.

However, planning for neither the whole track nor each action is hard for transferability. So, we devised a new framework to segment track to corners and plan trajectory separately.

We use model-free learning to plan key-point policy and use simple friction model to plan velocity limits. In low-level, the trajectory is generated by spline interpolate, and in high-level we use A\* graph search to minimize heuristic cost function. This hierarchical framework transfer policy and cost between two levels and iteratively optimize trajectory for very corner on the track.

We are working this method and will do more experiment for further development. Thanks for watching.

Great thanks for professor’s instruction and our group members Catherine, Wei, Liting, and Chen.