ESE 615 Final Project Proposal

Kedar Prasad Karpe Griffon McMahon Jiatong Sun

Autonomous racing by its very nature is a competitive field. At its core, it has one goal: for a vehicle to travel from the start to the finish faster than any other vehicles. These other vehicles often pose challenges, however, as they (obviously) also want to win. Therefore, potential controllers should not only aim to drive quickly, but they must also get ahead of the other cars. In this case, we will examine a Model Predictive Control (MPC) approach that greedily attempts to overtake a single opposing vehicle. We will apply this controller to an *F1TENTH* racecar [1].

Naturally, using MPC will require an understanding of the opposing vehicle's state over the controller's horizon. This is a challenging problem, especially because in a race, it is not natural to have the ground truth position and velocity of an opposing car. Thus, we will embark on the challenge of using lidar to provide a rough estimate of the opposing vehicle's state.

The aim of this project has two main parts. Firstly, we will implement an MPC controller. Secondly, we will implement an opponent state estimator. The exact process of developing these two will be discussed in section II-A.

I. Current Work

A no-frills implementation has been attempted [2] as discussed in class. This implementation only aims to drive quickly and does not appear to prioritize overtaking, instead treating opposing cars as mere obstacles.

A more recent paper [3], however, does something very similar to what we propose on the MPC front, though they do not claim it as a contribution. Perhaps they cite someone who does it more explicitly. Buyval et al. include the variable d^{σ}_{obs} (as in Fig. 1) in their state space for their nonlinear MPC, which is the distance between the egocar and the car to be overtaken, attempting to minimize it. They also implement a barrier function to avoid collision, which we will likely adopt. Opponent car prediction, however, is not within the scope of [3].

On the side of opponent prediction, one paper does propose an optimization-based approach to predicting an opponent's path [4], though this seems like overkill, especially given that we have a time crunch. In the field of drones, meanwhile, [5] use a simple linear interpolation of positions between points in time for velocity estimates. We will likely implement a similar feature.

II. Project Proposal and Timeline

Here we will discuss our plans for the project itself.

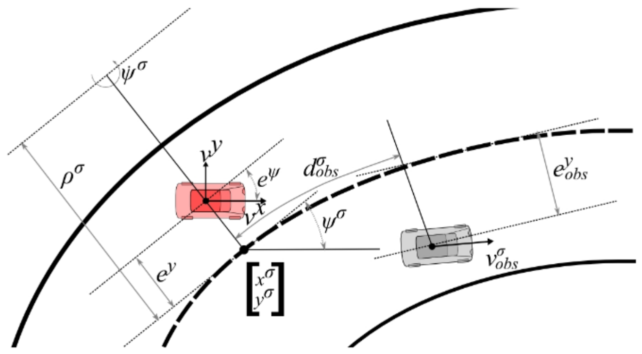

Fig. 1. A diagram showing state variables from [3].

A. Proposal

As stated earlier, our project has two parts: the MPC and the state estimation.

1) MPC: The MPC implementation is likely the easiest part of the project, especially as course TAs have offered to give starter code. Furthermore, the framework laid out in [3] will be valuable in implementation. How we will differ, however, is that instead of minimizing d^{σ}_{obs} as they do, we will instead attempt to maximize the relative d^{σ}_{obs} as an indicator of progress in a similar sense to the progress presented in [2]. The reference trajectory will be created using the minimum contour technique discussed in class.

2) Opponent Prediction: Our aim is to predict the opponent's state using only lidar. Currently, we plan on comparing the lidar scan with the map, after which we conclude that any scan that does not match the map must be an obstacle of some sort. We also know that there will only be one obstacle: the opposing car. Therefore, even in spite of mapping errors, we can find the cluster of the most number of points that differ from the map and label them as the other car. With this, we can collect the opponent car's position at various points in time. Then, we can use a linear interpolation of the positions over time to achieve a rough velocity estimate.

Assuming that the opponent vehicle is also following a semi-optimal trajectory, we can scale our own trajectory using their "known" velocity to create a rough estimate, mapping the opponent car's current velocity to our planned velocity at that point, their minimum observed velocity to our minimum planned velocity, and their maximum to our maximum. Alternatively, we can simply assume that the opponent car will continue in a straight line. Decisions will be made as implementation continues.

In the case that no opponent is detected (e.g., the opponent is behind the egocar), the MPC state variables that affect the decision will be set to zero.

B. Timeline

Fortunately, as the project has two parts, we can develop them in parallel. MPC will be developed and tested in simulation [6] using opponent ground truth until the opponent prediction is finished or usable. ROS nodes will work well for modular implementation here. This should be finished before the race.

As for opponent prediction, we will first aim at detecting opponent location with lidar. This is possible before the race, though more sophisticated prediction than a simple linear interpolation will be unlikely.

We have set up the timeline in Table I with the aim of having a rough implementation ready for the race on April 25th, though this is a tight schedule. In the event that we are not ready for the race, the testing and data gathering afterward can be compressed to allow further development time.

TABLE I
PROJECT TIMELINE AND DEADLINES

Date	MPC Goal	Opponent Prediction Goal
April 13th	Implement minimum contour	—
April 15th	Track reference trajectory with MPC	Estimate opponent location with simple velocity estimate
April 18th	Implement overtaking boundry function as in [3]	—
April 19th	Integrate both systems for the race	
April 25th	**Race 3**	
April 29th	Tuning as needed	Implement more sophisticated prediction
May 6th	Finish data collection	
May 9th	**Project Demonstration**	

C. Measuring Success

As a large part of the contribution is modifying the MPC's objective function, it is unknown what sort of performance will result. Therefore, we determine success as being able to pass an opponent car without crashing, with a secondary objective on doing so in a fast and efficient manner.

REFERENCES

[1] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio *et al.*, "F1/10: An open-source autonomous cyber-physical platform," *arXiv preprint arXiv:1901.08567*, 2019.

[2] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[3] A. Buyval, A. Gabdulin, R. Mustafin, and I. Shimchik, "Deriving overtaking strategy from nonlinear model predictive control for a race car," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2623–2628.

[4] R. Reiter, F. Messerer, M. Schratter, D. Watzenig, and M. Diehl, "An inverse optimal control approach for trajectory prediction of autonomous race cars," *arXiv preprint arXiv:2204.01494*, 2022.

[5] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.

[6] V. S. Babu and M. Behl, "f1tenth.dev - an open-source ROS based f1/10 autonomous racing simulator," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 1614–1620.