Implementing Nominal MPC on F1/10 racecar

1st Mitesh Agrawal
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
msagrawal@wpi.edu

1st Mohammed Tousif Zaman
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
mzaman@wpi.edu

***Abstract*—Safe navigation is one the biggest concerns in the research of autonomous vehicles. One of the best way to achieve this is through Model Predictive Control (MPC) of vehicle actuations. This paper discusses about various factors involved in the design and implementation of a closed loop control on a 1/10th scale F1 race car [II]. The paper also provides detailed problem statement and goals planned towards achieving this goal [III] [IV].**

***Index Terms*—Kinematic Bicycle Model, VICON State Estimation, Receding Horizon Control, Model Predictive Control, F1 Race Car, Self-driving car**

I. Introduction and Motivation

Today we are closer than ever to seeing the autonomous car become a reality. Internet giants like Google to automaker Nissan are working on autonomous technology, the streets may soon be filled with driverless cars [4]. A recent study conducted by Columbia University reports that replacing a fleet of 13,000 yellow cabs with 9,000 driverless cars in New York could cut costs per mile by nearly 88% and wait times down from 5 minutes to just 36 seconds during rush hour [5]. One of the biggest challenge for an autonomous vehicle is the tracking of a reference trajectory with accuracy to guarantee the safety of passengers as well as the surrounding environment comprising other vehicles on the road, pedestrians and objects in the path. For such systems when trajectories change at every time step depending on the changing conditions of the road, Model Predictive Control (MPC) seems to be the best possible solution. The essence of MPC is to optimise forecasts of process behaviors [3]. Though, a lot of research has been done in the application of MPC to self-driving cars, it is yet to achieve the required level of robustness. In this project, we first implement a controller on a $1/10^{th}$ scale F1 race car using Nominal MPC and a kinematic bicycle model. We then extend the implementation of Nominal MPC to Robust MPC to achieve higher speeds and still be able to avoid obstacles in the path.

II. Problem Statement

In this project we would like to understand how MPC could be implemented for trajectory tracking of an autonomous vehicle. We would like to understand in detail how the speed of the vehicle, limits on maximum deviation from equilibrium, change in trajectory and other constraints affect the performance of the system specially, robustness.

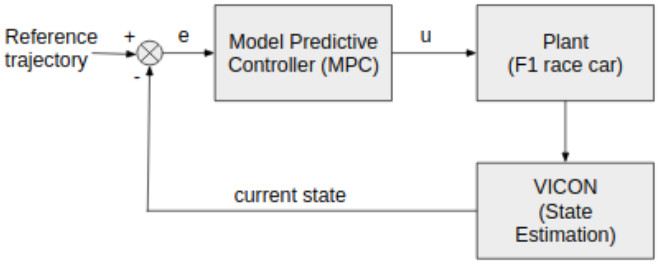

Fig. 1. Block diagram of the closed loop system

The race car must be able to track a given reference trajectory using MPC with state estimation given by the VICON system.

III. Goals

- Study the VICON system for state estimation
- Study Model Predictive Control (MPC) algorithm
- Set up ROS-Kinetic on the robot and base station
- Implement Nominal MPC for trajectory tracking on the 2D kinematic model of the 1/10th scale F1 race car

IV. Preliminaries

A. Kinematic Bicycle Model

To implement MPC on the race car, we will be using the non-linear kinematic bicycle model (Figure 2). In the bicycle model, only the front wheel can be steered, therefore, the control inputs are the front steering angle, δ_f and the rear steering angle, δ_r. x and y are the coordinates of center of mass of the vehicle, ψ is the inertial heading, β is the angle of current velocity of the center of mass with respect to the longitudinal axis of the car and v is the speed of the vehicle. lf and lr represent the distance from the center of the mass of the vehicle to the front and rear axles, respectively. The kinematic model of the race car can be viewed as below [1].

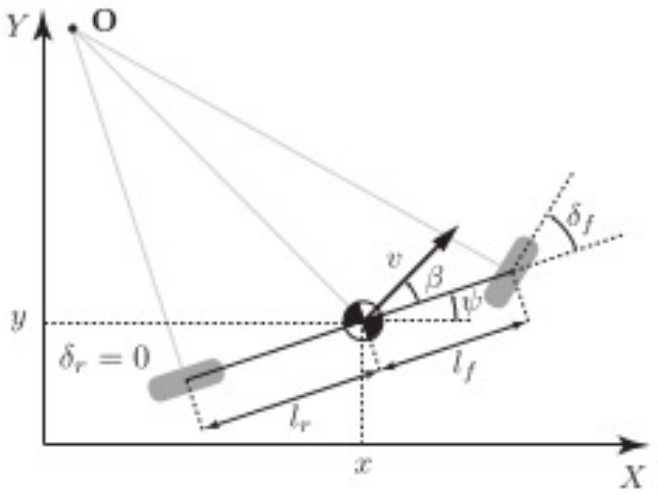

Fig. 2. Kinematic Bicycle Model.

$$\begin{aligned}
\dot{x} &= v\cos(\psi+\beta)\\
\dot{y} &= v\sin(\psi+\beta)\\
\dot{\psi} &= \frac{v}{l_r}sin(\beta)\\
\beta &= \tan^{-1}(\frac{l_r}{l_f+l_r}\tan(\delta_f))
\end{aligned} \tag{1}$$

$$z = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \tag{2}$$

$$u = \begin{bmatrix} \delta \\ v \end{bmatrix} \tag{3}$$

B. Model Predictive Control

MPC considers the trajectory tracking as an optimization problem which could be achieved in steps over a finite time horizon T subject to constraints on the system state and control input. The idea is that the controller predicts the future trajectories in advance and computes the error between reference and the predicted trajectories at each sampling time step until the end of the horizon. This prediction and control horizon are shifted each time step until the final goal is reached therefore it is also called Receding Horizon Control [7]. Assume $x(0)$ and $x(T)$ to be the initial and final states respectively, of the car and given a set of predicted trajectories, we obtain for each trajectory, a cost J that is a quadratic function of the error in the system state x and the control input u. The optimization problem is solved by minimizing the cost J and the corresponding predicted trajectory is chosen as the dynamically feasible trajectory. The equations of the linearized system, linear constraints and the quadratic cost function in case of Nominal MPC as explained in [7], [8] are:

$$\begin{aligned}
\min_u \ & \sum_{i=0}^{N-1}(z_i - z_{ref,i})^2 Q + \sum_{i=0}^{N-1}[(u_i - u_{ref})^2 R] + (z_N - z_{ref,N})^2 Q_f\\
\text{s.t. } & z_0 = z_t, u_{-1} = u_(t - t_s)\\
& z_{i+1} = f(z_i, u_i), i = 0, ..., N-1\\
& u_{min,i} \le u_i \le u_{max,i} \forall i\\
& y_{min,i} \le y_i \le y_{max,i} \forall i
\end{aligned} \tag{4}$$

V. Methodology

A. Discretization of Kinematic Bicycle Model

Consider the state space representation of the system as given in (5) where z represents the state as given by (2). The ODE is solved now by disretization using Euler approximation [9]. The state and input matrices (6),(7) here are obtained by Jacobian Linearization about the equilibrium point. The local equilibrium points are chosen to be on the reference trajectory. The resulting matrices are described in (8), (9) and are further used in the controller.

$$\dot{z} = Az + Bu \tag{5}$$

$$A = \begin{bmatrix} 0 & 0 & -v_0 sin(\psi+\beta) \\ 0 & 0 & v_0 cos(\psi+\beta) \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

$$B = \begin{bmatrix} 0 & cos(\psi+\beta) \\ 0 & sin(\psi+\beta) \\ \frac{v_0(1+tan^2(\delta))}{L_f+L_r} & \frac{tan(\delta)}{L_f+L_r} \end{bmatrix} \tag{7}$$

$$A = \begin{bmatrix} 1 & 0 & -v_{ref} sin(\psi) T_s \\ 0 & 1 & v_{ref} cos(\psi) T_s \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$B = \begin{bmatrix} 0 & cos(\psi) T_s \\ 0 & sin(\psi) T_s \\ \frac{v_{ref}(1+tan^2(\delta))}{L_f+L_r} T_s & \frac{tan(\delta)}{L_f+L_r} T_s \end{bmatrix} \tag{9}$$

B. State Estimation

In order to get the current state $z = (x, y, \psi)$ of the system, we will be using the VICON motion capture system. Though in real life scenarios, estimation is done using GPS, IMUs (Inertial Measurement Unit), RADAR (Radio Detection and Ranging) and LIDAR (Light Detection and Ranging) leading to ambiguity in estimated results necessitating the use of a Kalman filter which results in complex calculations. To avoid this, we consider only the VICON system for state estimation in this project [6].

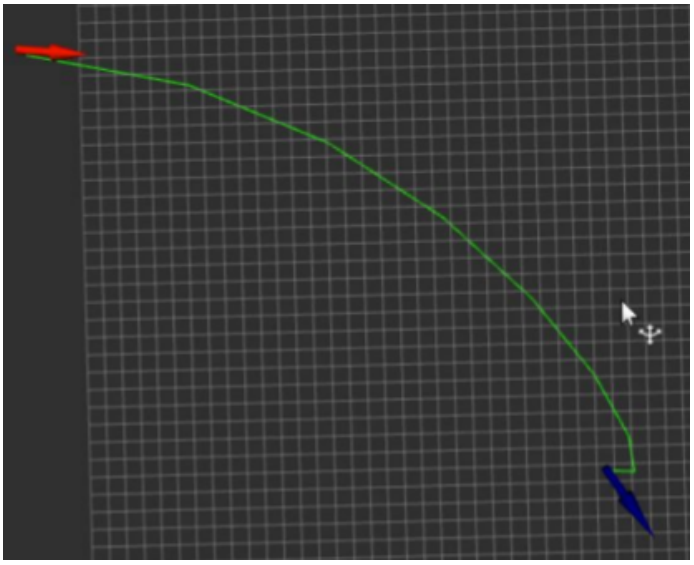

Fig. 3. Trajectory generation in RViz

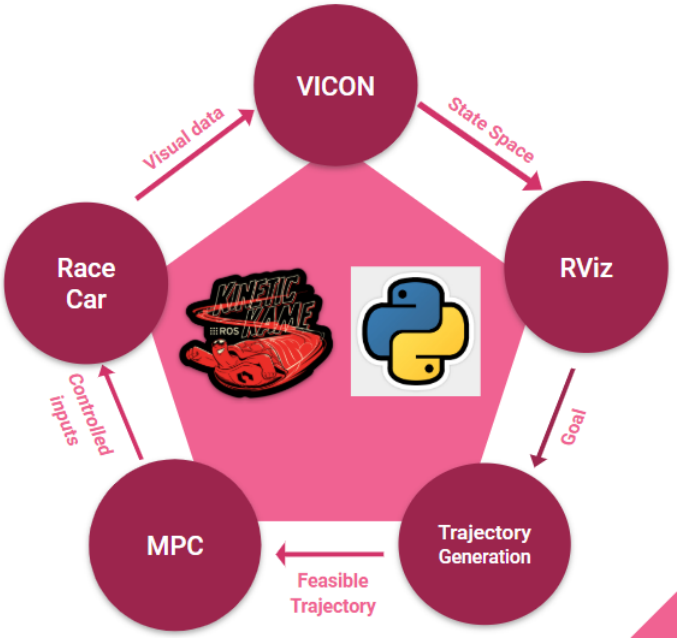

Fig. 4. Ros setup

C. Trajectory Generation

For this project we are only concerned about the trajectory tracking of the car i.e. reference trajectory (x_{ref}, u_{ref}) is provided as the input to the system. This reference trajectory is generated using cubic interpolation splines to connect the waypoints between the starting position and the goals. We will create cubic polynomials for x_{ref} and y_{ref} that are based on time as explained in [2]:

$$a_0 + a_1 t^1 + a_2 t^2 + a_3 t^3 = f_x(t) \tag{10}$$

$$b_0 + b_1 t^1 + b_2 t^2 + b_3 t^3 = f_y(t) \tag{11}$$

$$\dot{x}\sin(\theta) - \dot{y}\cos(\theta) = 0 \tag{12}$$

$$\dot{x}\cos(\theta) + \dot{y}\sin(\theta) = v \tag{13}$$

D. ROS Setup and Software

We used ROS Kinetic in Python language for our implementation. The code base maintained at [10] was forked and updated during the course of our project. The current implementation is now available at [11].

VI. Experiments and Results

A. Horizon and timestep matching

As mentined in earlier chapters we used the CVXPY optimisation library of python for optimisation in MPC. The time required for optimiser to generate results at each step is around 0.7 seconds. Which is actually very low when it comes to race car. The controller time step was thus decided to be 1 second. The system recived information about the state in every 1 second from the VICON pose estimation. It then calculated the MPC results and gave input to the motors. Thus the system was in sync with the timestep. It was observed that VICON testing area was having diagonal length of around 6 meters. With the car speed of one meters per second the maximum length of trajectory possible could be divided into 6 time steps. Thus our horizon was limited to 2 to 4 to see the behevior of the MPC on race car.

B. Tuning The Gain Matrices

The efficiency of MPC highly depends on the gain matrices defined for cost function of the controller, like any other controller. For tuning the MPC we started with the diagonal unit matrices in input cost matrix, state cost matrix as well as terminal cost matrix.gradually we tuned the input cost gain to provide relatively high penalty on the steer so that system outputs steer close to the reference steer. We observed that though system follows the similar steer it is not able to match the reference pose as desired. To elivate this effect we also increased the gain of pose in state cost. The effect of this was quite visible. The output were not now similar to our expectations. However, system showed bad response to minimize the error in states overall. We inreased the penalties on state cost overall and on terminal cost. This suddenly improved the results of MPC and gave a very good response to minimize error, following the trajectory pose as well as not deviating much from reference steer. Our final gain matrices are mentioned below.

$$R = \begin{bmatrix} 6 & 0 \\ 0 & 2 \end{bmatrix} \tag{14}$$

$$Q = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 10 \end{bmatrix} \tag{15}$$

$$Q_f = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{bmatrix} \tag{16}$$

Overall MPC provides large range if flexibility to modify gain matrices to match our requirements and priorities. In order to match any other requiremts one can modify the cost matrix and tune it to get optimal solutions. The tuning of gain matrix change from system to system based on the dynamics of system, optimiser used, horizon of MPC and length of time step.

C. Hard Constraint On Velocity Input

The CVXPY library used for the optimization of MPC allows us to define only soft constrains. The MPC was given constraints of minimum and maximum states as well as input. Optimiser respected the constraints on states and steer very well but it was seen that the optimisers ignore constraints on velocity sometimes. This resulted in high difference in velocities at each timestep as well as negetive velocities being given as input. High velocities lead to large diviations from planned trajectories. This also ment that system was at high risk of moving out of vicon region and hitting the walls. Also car was not designed to execute the negetive velocities published at some points. Thus hard constraint on velocity was imposed after the optimiser result. We saturated the optimiser output velocity between zero and maximum velocity to avoid any damage to car.

D. Low Speed Performance

After timestep matching and tuning of the gains the car showed very satisfactory results in tracking the given trajectory. The system was first tested on low velocities to allow larger number of timesteps as well as horizon to see the performance. The speed of 0.5 meters per second was set as maximum limit in the constraints of MPC. For straight line trajectories the system worked very well. Figure below shows the performance of MPC on straight line trajectory. MPC delt with errors very smoothly and recovered the trajectory. There is no lagging with the timestep as well as the system respected reference pose and steer very well as expected based on input penalties in cost function. We also checked for small distubances kept in the path of the car. The car was able to come back to its trajectory.

Curved paths are more complex in case of race car due to relatively high speed. MPC traced the curved path very well. It was able to take smooth curve with stability, no drag, in the path. The performance of the MPC on curved path be seen in the graph below. The system is observed to take longer to minimize error but it deals with the errors very smoothly. We can observe very well from the graph that system is following pose of the trajectory even after diviating from the trajectory and tries to minimize error gradually.

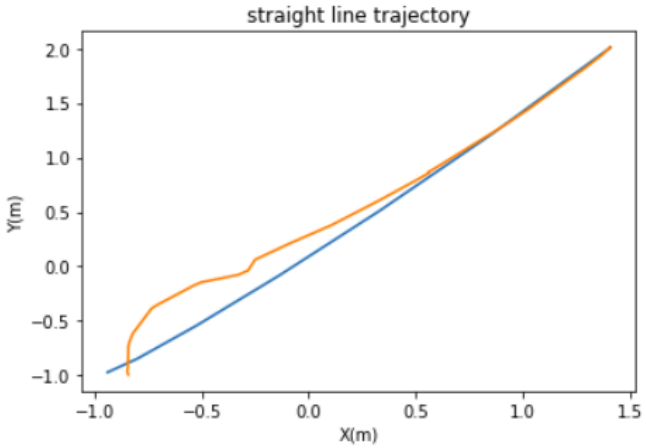

Fig. 5. low speed performance on straight line(blue line: trajectory given; Orange line: trajectory tracked).

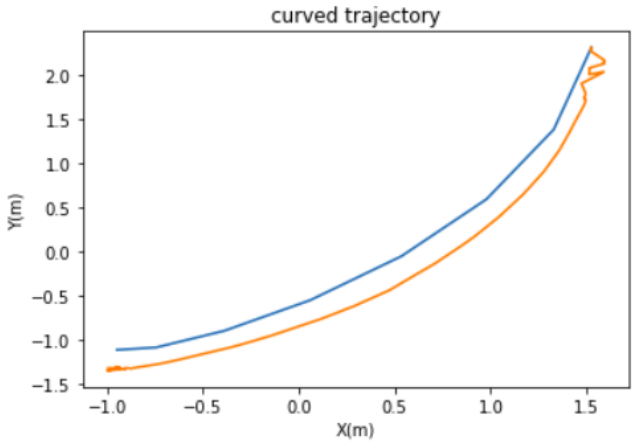

Fig. 6. low speed performance on curved line(blue line: trajectory given; Orange line: trajectory tracked).

E. High Speed Performance

Speed is the most important factor for race car. That being said the performance of MPC on high speed tracks are vital to be tested before connenting on the efficiency of the controller installed on it. We checked the performance of racecar on maximum limit of 3 meters per second. As the optimization time of our optimiser was high the high speed performance was already expected to be lower than low speed performance. This is because the distance travved in each time step is high (approximately 3 meters) leading to high error at every step. Also as mentined earlier the playgroud of VICON was having diagonal of 6 meters the was able to make it upto only 3 time steps before completing the trajectory, which is very low to see the performance of MPC overall. The performance of MPC on curved paths can seen below. The system was observed to minimize error at every timestep but the playground area is certainly smaller to allow MPC to minimize the error to zero.

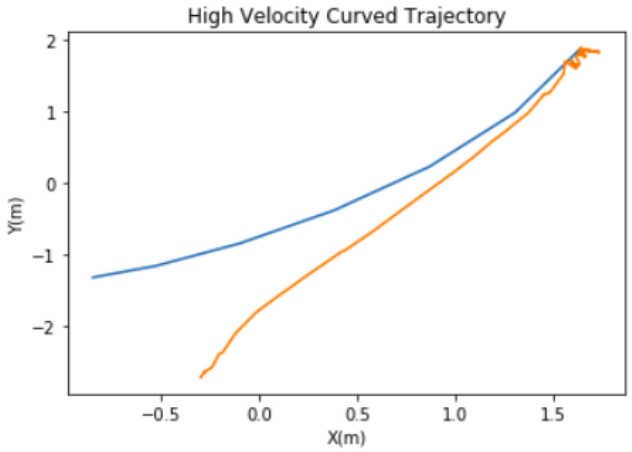

Fig. 7. high speed performance on curved line(blue line: trajectory given; Orange line: trajectory tracked).

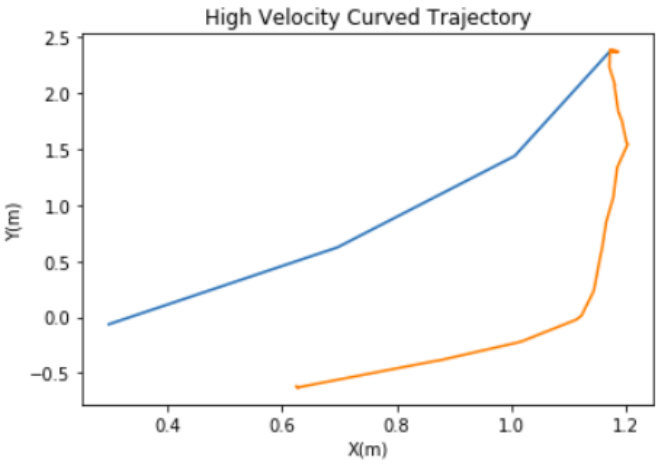

Fig. 8. high speed performance on curved line(blue line: trajectory given; Orange line: trajectory tracked).

In order get better performance in high speed better optimiser and processing speed is required to reduce the length of time step allowing precision in controller.

Also in the case of high speed permance we limited our horizon to 2. This is because the trajectory comprised on only 3 to 4 timesteps. This increased the performance of MPC in high speed tracking.

F. Difficulties

One of the biggest challenge while dealing with MPC is matching the timestep with real time. The sampling of statespace to 1 second and setting the time step to 1 second solved this problem. But this solution is not feasible in real world. In case the system respoce to too slow with initial inputs of lower velocity, the system is observed to lag or overshoot the timestep which it is supposed to be at. In cases of lagging the optimiser is able to continue till it completes the trajectory but in the case of overshooting the system is confused regarding the next input of the car and optimiser fails leading to abrupt outputs in steer as weel as velocity leading to deviation from the trajectory. This is still an open point that needs to be solved in the system.

VII. Future Scope

Upgrading system to a better optimiser with lower time of response will be the first step to improve the performance of the MPC. MPC also provides a large range of flexibility to play with cost functions. Including better factors to control on high speed track may help improving its performance manifolds. Another step will be to implement robust MPC to allow error minimization in lesser time as well as sharp curves. In order to alleviate the effect of time step overshoot and lag we can improvise the system to automatically decide its current time step based on nearest point on the trajectory.

VIII. Conclusion

The performance of MPC depends on many factors. The first is the length of the timestep which in turn depends on the optimizer used. A fast optimizer will lead to lower time step. This will lead to precision in the control of the system. In general, larger the horizon of the MPC, more smooth is the performance of the system.

The optimization cost function is the heart of MPC which controls its overall behevior. MPC provides great flexibility to play with cost functions to match our requirements.

References

[1] J. Kong, M. Pfeiffer, G. Schildbach, F. Borrelli, Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design, in Proceedings of the IEEE Intelligent Vehicle Symposium, pp. 1094-1099, June 2015.
[2] J. Chen, A. Huynh, Z. Jiang, Z. Wu. Self Driving on 1/10 Racer Car, Worcester Polytechnic Institute. 2018.
[3] J. Rawlings, Tutorial overview of Model Predictive control, IEEE control systems Magazine, June 2000.
[4] Benjamin Zhang. Autonomous cars could save the us 1.3 trillion dollars a year. Business Insider, September 2014.
[5] John Markoff, Googles Next Phase in Driverless Cars: No Steering Wheel or Brake Pedals. The New York Times, May 2014
[6] Pierre Merriaux, Yohan Dupuis, Rmi Boutteau, Pascal Vasseur, Xavier Savatier. A Study of Vicon System Positioning Performance. Sensors, MDPI, 2017, 17 (7), 10.3390/s17071591.hal-01710399
[7] Model Predictive Control of Hybrid Systems of Hybrid Systems by Alberto Bemporad; http://cse.lab.imtlucca.it/ bemporad/hybrid/school07/pdf/08.Bemporad.pdf
[8] Desaraju, Vishnu R., "Safe, Efficient, and Robust Predictive Control of Constrained Nonlinear Systems" (2017). Dissertations. 954. http://repository.cmu.edu/dissertations/954
[9] B. N. BISWAS, S. CHATTERJEE, S. P. MUKHERJEE AND S. PAL, A DISCUSSION ON EULER METHOD: A REVIEW
[10] https://arcgit.wpi.edu/vishnurs/rccar-demo
[11] https://arcgit.wpi.edu/mzaman/rccar-demo