Designing a Receding Horizon Planner for an Autonomous Formula Student Racecar

Kerry He
Supervised by Hoam Chung

***Abstract*—In autonomous driving, the motion planning subsystem is required to determine a feasible state and control trajectory to navigate the vehicle to perform a specific task. This paper presents an implementation of a receding horizon planner (RHP) to perform motion planning for Monash Motorsport's autonomous racecar to compete in the Formula Student Driverless competition. Multiple modelling and discretisation methods were explored to determine the best performing RHP formulation. Ultimately, a linear time-varying RHP formulation utilising a dynamic bicycle model is proposed, where the vehicle dynamics and path constraints are linearised at each time step, allowing the RHP to be formulated and solved as a quadratic program. Through simulated experiments, the proposed RHP is shown to successfully outperform Monash Motorsport's previous motion planning implementations, and is demonstrated to safely achieve speeds of up to 25 m/s while running in real time at 50 Hz.**

Fig. 1. Monash Motorsport's driverless car, M19-D, driving autonomously.

I. Introduction

The growth of autonomous vehicle technology has been accompanied with an interest in its application in racing, with autonomous racing competitions such as the DARPA Grand Challenge [1], Formula Student Driverless (FSD)[1] and Roborace[2] growing in popularity. A key sub-system in autonomous driving software architectures which is pivotal to maximising the racing capabilities of autonomous vehicles is motion planning, which aims to find a dynamically feasible trajectory, then follow this trajectory in real time in the presence of noise, disturbances, and other uncertainties.

Several motion planning algorithms for autonomous racing have been proposed in literature, such as pure pursuit [2] and Stanley control [1]. An increasingly popular class of motion planners are optimal control techniques such as receding horizon planning (RHP) [3]–[5]. Compared to other algorithms, the greatest advantage of optimal control techniques are their unique ability to directly incorporate state and control constraints into the problem formulation in real time. Moreover, optimal control problems can be formulated using high-fidelity vehicle models, such as the kinematic bicycle model or the dynamic bicycle model [3]. These properties are particularly advantageous for autonomous racing, where maximising lap time performance requires a thorough understanding of the vehicle dynamics and handling limits of the vehicle.

Optimal control problems can be solved numerically by discretising the problem into an optimisation problem, which is then solved using suitable optimisation algorithms. However, this can be prohibitive for complex or nonlinear systems, such as the bicycle vehicle models, as the resulting nonlinear program can be difficult to solve in real time. A common approach to improve the real-time feasibility of the RHP is to linearise the vehicle dynamics and constraints at each time step to achieve a linear time-varying RHP (LTV-RHP), which allows the problem to be formulated and solved more easily as a quadratic program (QP) [4], [5].

Monash Motorsport (MMS) is Monash University's Formula Student team which developed Australia's first autonomous Formula student racecar, M19-D, with the aim to compete in FSD. M19-D is currently using an early RHP implementation as its motion planner, which outputs target velocities and steering wheel angles which are tracked by low-level PID controllers. Due to being an early implementation, the main goal for the planner was to successfully finish missions, with lap time performance only being a secondary subgoal. Therefore, the RHP was implemented using a relatively simplistic formulation, and an optimal control library, ACADO Toolkit [6], was used as a high-level interface to solve the RHP problem.

This paper improves upon M19-D's previous RHP implementation to achieve faster lap times by exploring two key facets to the formulation of an RHP. Firstly, the paper improves upon the modelling used in the RHP by investigating more complex vehicle models and constraint formulations. Secondly, to achieve additional flexibility in the RHP formulation, discretisation is performed manually as opposed to relying on an optimal control library.

II. Vehicle Model

The racecar is modelled using a dynamic bicycle model shown in Figure 2, which is derived by considering tyre forces

[1]https://www.formulastudent.de/fsg/
[2]https://roborace.com/

and the resulting kinetics of a single-track model about its centre of gravity (CoG). Longitudinal tyre forces arising from slip from the front tyres and aerodynamic forces are assumed to be negligible. The vehicle is defined by its mass m, yaw moment of inertia I, and dimensions from its CoG l_f and l_r.

The vehicle state is defined in curvilinear coordinates, which defines the vehicle relative to a specified path by introducing the arclength travelled along the path s, the lateral deviation from the path n, and the angular deviation from the path μ. The path is constructed using piecewise cubic splines, and is characterised by its curvature $\kappa(s)$. Additionally, the vehicle state includes the longitudinal $\dot{x}$ and latitudinal $\dot{y}$ velocities, defined in the vehicle's inertial frame, the yaw rate $\dot{\theta}$, and the front steering wheel angle δ.

The control variables are defined such that the steering rate $\dot{\delta}$ and driving force F_x are modelled as a first order transient response to account for nontrivial actuator transient delay. Thus, the control variables are defined as the velocity v_t and steering wheel angle δ_t target setpoints. K_v and K_δ represent the equivalent gains of a proportional controller.

Overall, the vehicle model dynamics are expressed as

$$\dot{s} = \frac{\dot{x}\cos(\mu) - \dot{y}\sin(\mu)}{1 - n\kappa(s)} \tag{1a}$$

$$\dot{n} = \dot{x}\sin(\mu) + \dot{y}\cos(\mu) \tag{1b}$$

$$\dot{\mu} = \dot{\theta} - \frac{\dot{x}\cos(\mu) - \dot{y}\sin(\mu)}{1 - n\kappa(s)}\kappa(s) \tag{1c}$$

$$\ddot{x} = \frac{1}{m}(mK_v(v_t - \dot{x}) - F_{cf}\sin(\delta) + m\dot{y}\dot{\theta}) \tag{1d}$$

$$\ddot{y} = \frac{1}{m}(F_{cr} + F_{cf}\cos(\delta) - m\dot{x}\dot{\theta}) \tag{1e}$$

$$\ddot{\theta} = \frac{1}{I}(l_f F_{cf}\cos(\delta) - l_r F_{cr}) \tag{1f}$$

$$\dot{\delta} = K_\delta(\delta_t - \delta). \tag{1g}$$

Using the Pacejka Magic Formula [7], the front and rear lateral tyre forces are defined as functions of the slip angle and normal force on each tyre

$$F_{cf} = f_c(\alpha_f, F_{zf}) \tag{2a}$$

$$F_{cr} = f_c(\alpha_r, F_{zr}), \tag{2b}$$

where the slip angles of the tyres are defined as

$$\alpha_f = \delta - \arctan\left(\frac{\dot{y} + l_f\dot{\theta}}{\dot{x} + \dot{x}_{min}e^{-\dot{x}/\dot{x}_{min}}}\right) \tag{3a}$$

$$\alpha_r = -\arctan\left(\frac{\dot{y} - l_r\dot{\theta}}{\dot{x} + \dot{x}_{min}e^{-\dot{x}/\dot{x}_{min}}}\right). \tag{3b}$$

The slip angle equations are augmented with a term in the denominator which decays with the vehicle's longitudinal velocity, which is included to avoid the tyre forces from becoming singular at low velocities.

The weight distribution between the front F_{zf} and rear F_{zr} tyres is estimated using a simple constant moment balance about the centre of gravity. Note that this simplification ignores load transfer effects typically experienced by racecars.

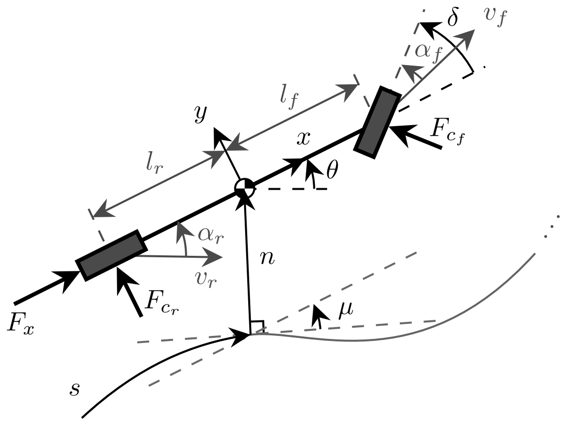

Fig. 2. Dynamic bicycle model in curvilinear coordinates

III. Receding Horizon Planning

Consider the nonlinear system $f\colon \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\xi}$ defined by the dynamic bicycle model presented in Section II

$$\dot{\xi}(t) = f(\xi(t), u(t)), \tag{4}$$

where $\xi = [\, s \;\, n \;\, \mu \;\, \dot{x} \;\, \dot{y} \;\, \dot{\theta} \;\, \delta \,]^T$ and $u = [\, v_t \;\, \delta_t \,]^T$ represent the state and control vectors respectively. The RHP is formulated as a reference tracking optimal controller, such that the objective function is defined as

$$\begin{aligned} J(\xi, u, \epsilon, t_0) = {} & \|\xi(t_f) - \xi_r(t_f)\|^2_{Q_f} \\ & + \int_{t_0}^{t_f} \|\xi(\tau) - \xi_r(\tau)\|^2_Q + \|e(\tau)\|^2_R \, d\tau + q^T\epsilon, \end{aligned} \tag{5}$$

where $\xi_r \in \mathbb{R}^{n_\xi}$ is the state reference trajectory obtained from a precomputed optimal racing line, $e(t) = [\, F_x(t) \;\, \dot{\delta}(t) \,]^T$ represents the effort exerted by the vehicle, $\epsilon \in \mathbb{R}^{n_\epsilon}_+$ is a vector of slack variables, and Q, $Q_f \in \mathbb{R}^{n_\xi \times n_\xi}$, $R \in \mathbb{R}^{n_u \times n_u}$ and $q \in \mathbb{R}^{n_\epsilon}$ are state, terminal state, control, and slack variable weights respectively. The objective function is constructed to penalise deviations from the reference trajectory, while an effort minimising regularisation term is included to have a smoothing effect on the solution trajectory. A terminal penalty is included to help enforce stability [8].

Path constraints $g(\xi(t), u(t), \epsilon)\colon \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\epsilon}_+ \to \mathbb{R}^{n_g}$ consisting of a combination of state, control and functional constraints shown in (6) are enforced. Upper and lower bounds on quantities are expressed as $\overline{(\cdot)}$ and $\underline{(\cdot)}$ respectively.

$$\underline{n} - \epsilon_n \le n(t) \le \overline{n} + \epsilon_n \tag{6a}$$

$$\underline{\delta} \le \delta(t) \le \overline{\delta} \tag{6b}$$

$$0 \le \dot{x}(t) \tag{6c}$$

$$\underline{\Delta\delta} \le K_\delta(\delta_t(t) - \delta(t)) \le \overline{\Delta\delta} \tag{6d}$$

$$\left(\frac{K_v(v_t(t) - \dot{x}(t))}{\overline{a_l}}\right)^2 + \left(\frac{F_{cr}(t)}{m\overline{a_c}}\right)^2 - \epsilon_a \le 1 \tag{6e}$$

These constraints correspond to the track boundary constraints, steering angle limits, prevention from backwards movement, slew rate of the steering angle, and tyre friction ellipse constraints respectively. Constraints (6a) and (6e) are formulated as soft constraints to allow for an additional degree of flexibility in situations when the problem would otherwise be infeasible, and no solution exists which would satisfy the constraints. These are implemented using the slack variables $\epsilon = [\,\epsilon_n \;\; \epsilon_a\,]^T$, which are penalised using a minmax approximation of an ∞-norm penalty function [9].

Overall, the continuous-time RHP problem is formulated as

$$\begin{aligned}
&\min_{\xi,u,\epsilon} && J(\xi,u,\epsilon,t_0) && \text{(7a)}\\
&\text{subj. to} && \xi(t_0)=\hat{\xi} && \text{(7b)}\\
& && \dot{\xi}(t)=f(\xi(t),u(t)), \qquad t_0\le t\le t_f && \text{(7c)}\\
& && \underline{g}\le g(\xi(t),u(t),\epsilon)\le\overline{g}, \qquad t_0\le t\le t_f && \text{(7d)}\\
& && \epsilon\ge 0. && \text{(7e)}
\end{aligned}$$

Once the RHP (7) has been discretised using the method described in Section IV, it is repeatedly solved in a receding horizon fashion. This algorithm is visualised in Figure 3.

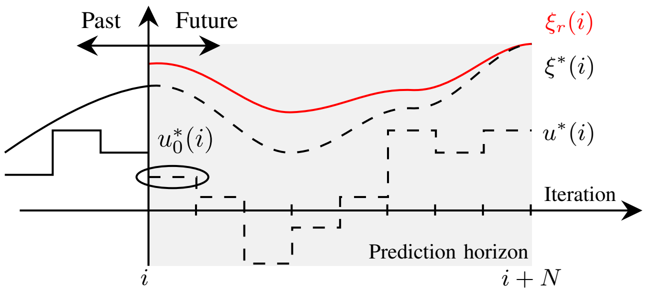

Fig. 3. Receding horizon planning algorithm. Only the controls at the first time step of the solved optimal control trajectory $u_0^*(i)$ are used. In the next time step, the prediction horizon shifts forward by a time step, and the optimal trajectory $\xi^*(i), u^*(i)$ is recomputed to find an updated set of controls.

IV. Successive Linearisation Discretisation

To solve the RHP, the time horizon is discretised into N discrete intervals spaced evenly by a time step Δt to achieve a finite number of optimisation variables that can be solved for. This results in a state trajectory vector $\xi = [\,\xi_1 \;\; \xi_2 \;\; \dots \;\; \xi_N\,]^T$ and control trajectory vector $u = [\,u_0 \;\; u_1 \;\; \dots \;\; u_{N-1}\,]^T$.

An LTV sequential single-shooting scheme is then used to discretise the RHP problem. This involves expressing the optimisation variables purely as the control trajectory u, which requires the state trajectory ξ to be approximated as a linear function of u. This results in the QP

$$\begin{aligned}
&\min_{u_\epsilon} && \frac{1}{2}u_\epsilon^T H u_\epsilon + f^T u_\epsilon && \text{(8a)}\\
&\text{subj. to} && \xi_0=\hat{\xi} && \text{(8b)}\\
& && \underline{\gamma}\le G u_\epsilon\le\overline{\gamma} && \text{(8c)}\\
& && \epsilon\ge 0, && \text{(8d)}
\end{aligned}$$

where the control vector and slack variables are combined into a single optimisation variable $u_\epsilon = [\,u \;\; \epsilon\,]^T$.

To achieve the QP formulation (8), consider the problem of transforming the objective function into a quadratic function solely in terms of u_ϵ. By introducing a padding $0_\epsilon \in \mathbb{R}^{n_\epsilon}$ as a zero vector with a length equal to the number of slack variables, and redefining the error vector correspondingly as $e_\epsilon = [\,e \;\; 0_\epsilon\,]^T$, the discretised objective function can be expressed as the matrix equation

$$J(\xi,u_\epsilon) = (\xi-\xi_r)^T\tilde{Q}(\xi-\xi_r) + e_\epsilon^T\tilde{R}e_\epsilon + \tilde{q}^T u_\epsilon, \qquad (9)$$

where

$$\begin{aligned}
\tilde{Q} &= \text{blockdiag}(Q,\dots,Q,Q_f) && \text{(10a)}\\
\tilde{R} &= \text{blockdiag}(R,\dots,R,0_\epsilon) && \text{(10b)}\\
\tilde{q} &= [\,0 \;\; \dots \;\; 0 \;\; q\,]^T. && \text{(10c)}
\end{aligned}$$

Now consider the Runge-Kutta fourth-order (RK4) integration scheme $f_{\text{RK4}}\colon \mathbb{R}^{n_\xi}\times\mathbb{R}^{n_u}\to\mathbb{R}^{n_\xi}$ used to integrate the plant model f

$$\xi_{k+1} = \xi_k + f_{\text{RK4}}(\xi_k,u_k). \qquad (11)$$

This can be linearised by taking the first order Taylor series expansion about a linearisation trajectory $\xi_l = [\,\xi_{l,0} \;\; \dots \;\; \xi_{l,N-1}\,]^T$, $u_l = [\,u_{l,0} \;\; \dots \;\; u_{l,N-1}\,]^T$, defined as the previously solved optimal state and control trajectory. The linearised equation can be expressed as the affine expression

$$\xi_{k+1} = A_k\xi_k + B_k u_k + d_k, \qquad (12)$$

where

$$A_k = I + \left.\frac{\partial f_{\text{RK4}}}{\partial \xi}\right|_{\substack{\xi=\xi_{l,k}\\ u=u_{l,k}}}, \qquad B_k = \left.\frac{\partial f_{\text{RK4}}}{\partial u}\right|_{\substack{\xi=\xi_{l,k}\\ u=u_{l,k}}} \qquad (13a)$$

$$d_k = \xi_{l,k} + f_{\text{RK4}}(\xi_{l,k},u_{l,k}) - A_k\xi_{l,k} - B_k u_{l,k}. \qquad (13b)$$

The partial derivatives of the RK4 scheme can be found in [10].

Once the system has been linearised, the state trajectory ξ needs to be formulated as a function of the control trajectory u_ϵ. This can be done by iteratively applying (12) from a given initial condition ξ_0. This can be expressed as

$$\xi = \tilde{A}\xi_0 + \tilde{B}u_\epsilon + \tilde{d}, \qquad (14)$$

where $\tilde{A}$, $\tilde{B}$ and $\tilde{d}$ are derived using a similar method to the batch approach method to solve linear quadratic regulators [11].

A complication arises as the effort term e_ϵ needs to be expressed as a difference between control and state variables, which can be difficult to do using a sequential discretisation scheme. Instead, a simplification is made such that e_ϵ is defined relative to the initial state for the entire horizon.

$$e_\epsilon = [\,u \;\; 0_\epsilon\,]^T - u_0, \quad u_0 = [\,\dot{x}_0 \;\; \delta_0 \;\; \dots \;\; \dot{x}_0 \;\; \delta_0 \;\; 0_\epsilon\,]^T \qquad (15)$$

Finally, given (14) and (15), the quadratic objective function (8a) can be derived such that

$$\begin{aligned}
H &= 2(\tilde{B}^T\tilde{Q}\tilde{B} + \tilde{R}) && \text{(16a)}\\
f &= 2(\tilde{B}^T\tilde{Q}(\tilde{A}\xi_0 + \tilde{d} - \xi_r) - \tilde{R}u_0) + \tilde{q}. && \text{(16b)}
\end{aligned}$$

The path constraints g can be similarly treated by linearising the constraint function along the same linearisation trajectory

ξ_l, u_l to find expressions for $G, \underline{\gamma}$ and $\overline{\gamma}$. In addition, to improve the robustness of the linearised RHP, the tyre friction ellipse constraint (6e) was linearised beforehand to create a convex set using N_a linearised constraints [5], and the front and rear slip angles of the tyres were softly constrained to ensure the vehicle remained within the linear regions of the lateral tyre force function [4].

V. RHP Formulation Analysis

Before evaluating the RHP performance in a full simulation, its performance was compared to alternative modelling and discretisation methods to justify the proposed RHP specifications. Comparisons were made against the kinematic bicycle model, and against nonlinear discretisation schemes including multiple shooting and trapezoidal collocation. These experiments were performed in MATLAB using a simplified simulation environment. Three tracks used in previous official FSD competitions including Formula Student Germany (FSG2019), Spain (FSS2019) and Online (FSO2020) were used to evaluate the planner performances. The tradeoffs between lap time performance and computation times between various RHP formulations are presented in Table I and Figure 5, where the results using the proposed formulation are highlighted.

The dynamic bicycle model (Dyn.) saw significant lap time advantages over the kinematic bicycle model (Kin.). As maximisation of tyre forces is one of the most important factors in vehicle racing [12], this lap time improvement is primarily attributed to the improved utilisation of the friction ellipse by being able to accurately model and constrain the tyre forces, as seen in Figure 4. Furthermore, although the kinematic bicycle model solved faster, the key requirement for real-time feasibility of the RHP is for the solve times to be less than the discretised time step, which was 50 ms in these preliminary experiments. Improvements to the RHP algorithm such as hot-starting and converting the code to C++ were expected to reduce the computation times by approximately a factor of 10, and bring the the dynamic bicycle model's computation times

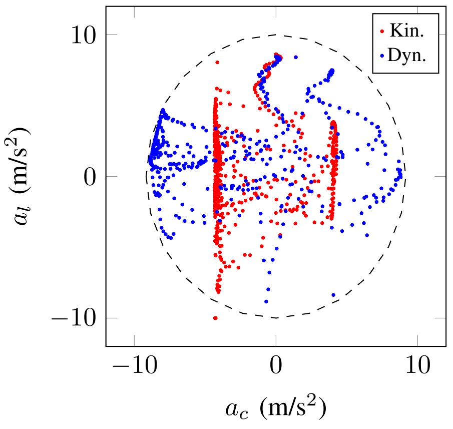

Fig. 4. Comparison of G-G plots over the first lap of FSG2019. The simulated friction ellipse is shown in dashed lines.

TABLE I
Comparison of single lap times between different RHP schemes.

Model	Discretisation	Total Lap Time (s)		
		FSG2019	FSS2019	FSO2020
Kin.	LTV	27.27	28.13	35.27
Dyn.	**LTV**	**22.43**	**22.17**	**27.70**
Dyn.	TC	21.90	22.05	27.55
Dyn.	MS	21.89	21.43	27.34

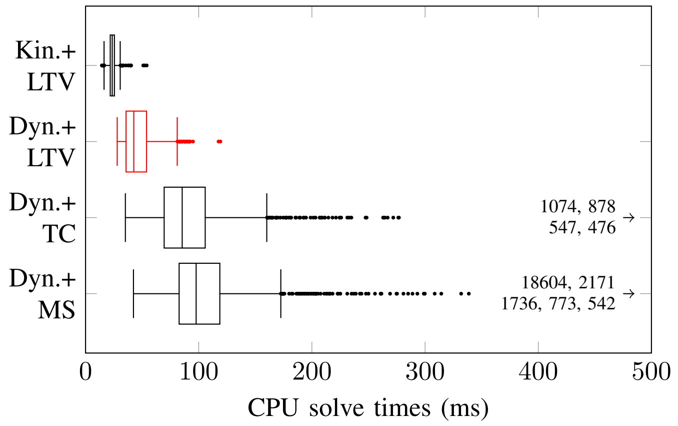

Fig. 5. Comparison of computation times of RHP schemes throughout simulated experiments of one lap of each of FSG2019, FSS2019 and FSO2020.

under this threshold. Therefore, the dynamic bicycle model was chosen for its superior lap time performance.

As the trapezoidal collocation (TC) and multiple shooting (MS) schemes were solved as nonlinear programs, they were observed to have significantly higher mean and maximum computation times compared to the LTV scheme. Although it is possible to set a maximum CPU time limit in optimisation solvers, stopping the optimiser before convergence means that feasibility of the optimal control solution is not guaranteed, and thus degrades the real time performance of the planner. Moreover, despite theoretically being more accurate by capturing the nonlinearities of the vehicle model, these nonlinear schemes did not achieve significantly improved lap times. Hence, the LTV scheme was chosen for being a much lighter formulation, and more suitable for online applications.

VI. Experimentation

A. Experimental Setup

A hardware-in-the-loop simulation environment shown in Figure 6 was used to evaluate the performance of the proposed LTV-RHP when fully integrated into MMS's autonomous systems pipeline. The commercial vehicle modelling software IPG CarMaker was used to simulate the vehicle dynamics.

The LTV-RHP was implemented in C++, and was solved using qpOASES [13]. Hot-starting and preset settings optimised for RHP problems were used. A dead time of 150 ms was compensated for by using a forward state prediction method similar to that described in [14]. A time step of $\Delta t = 20\,\mathrm{ms}$ with $N = 20$ was used.

The LTV-RHP was compared against two other motion planners which were previously used by MMS. The first is a baseline (BL) RHP formulation which utilises a kinematic bicycle model in Cartesian coordinates, has a time step of $\Delta t = 600\,\mathrm{ms}$ with $N = 5$, and is discretised and solved using ACADO Toolkit [6]. The second planner is a pure pursuit algorithm (PP).

All simulations were run on an Intel i7-6500U CPU running at 2.5 GHz with 8 GB of RAM on Linux Ubuntu 18.04. The same tracks used in Section V were used in these experiments.

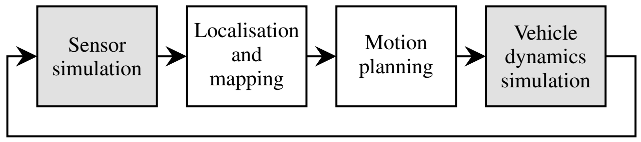

Fig. 6. Hardware-in-the-loop simulation environment pipeline used for final evaluation of the RHP. Grey blocks represent simulated components of the real world equivalents, and white blocks function identically between simulated and real world missions.

B. Racing Performance

The simulated lap time results are summarised in Table II, and the racing trajectory of LTV-RHP in FSG2019 is presented in Figure 9. Overall, LTV-RHP consistently and significantly outperformed all other planners in all tracks.

TABLE II
COMPARISON OF MISSION TIMES OVER 10 LAPS BETWEEN DIFFERENT PLANNERS.

	Total Lap Time (s)		
	FSG2019	FSS2019	FSO2020
LTV-RHP	**215.4**	**231.5**	**339.4**
Baseline RHP	307.7	282.9	476.9
Pure Pursuit	288.9	354.7	395.4

The improved performance of LTV-RHP can be attributed to a few key factors. Firstly, by using the more complex dynamic bicycle model, LTV-RHP was able to exploit the vehicle dynamics much more effectively, as discussed in Section V. As shown in the G-G plot in Figure 7, LTV-RHP was observed to drive at the edges of the friction ellipse for the majority of the track as opposed to the other two planners.

Secondly, LTV-RHP had improved path tracking capabilities due to the use of curvilinear coordinates and enforcement of track boundary constraints. BL and PP were both significantly hindered by the lack of these factors; the only tunable parameter for these planners to prevent track boundary violations was to reduce the maximum reference velocities and accelerations, as seen in the velocity plot in Figure 8. On the other hand, LTV-RHP skirted about the boundaries and corner-cut in a controlled manner much more frequently compared to BL and PP as seen in the lateral deviation plot in Figure 8,

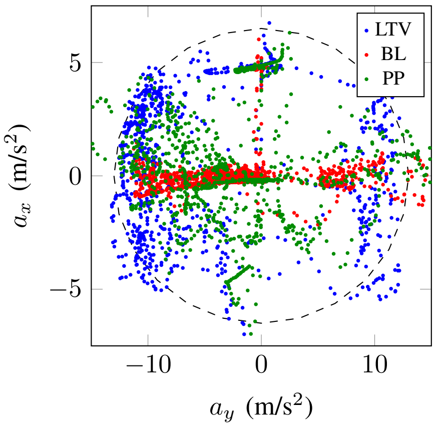

Fig. 7. Comparison of G-G plots over the first lap of FSG2019. Estimation of the friction ellipse is shown in dashed lines.

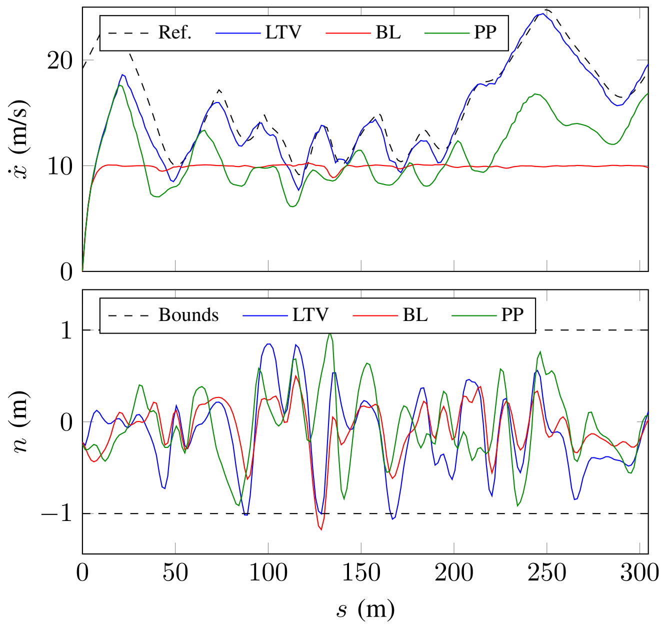

Fig. 8. Comparison of the longitudinal velocity (top) and lateral deviation (bottom) over the first lap of FSG2019 each planner.

despite travelling at much higher speeds and accelerations. PP was also observed to exhibit undesirable fluctuations in the lateral deviation arising from oscillatory steering as opposed to efficient corner cutting.

Finally, modelling the actuator transient response to help compensate for time delay significantly contributed to the improved performance. BL employed large time steps of $\Delta t = 600\,\mathrm{ms}$ to compensate for time delay. However, this led to overdamped controls which had trouble performing complex manoeuvres, which manifested as a flat velocity profile as seen in Figure 8. In comparison, by modelling the transient response, LTV-RHP could use much smaller time steps while remaining robust to time delay, allowing the planner to perform more aggressive manoeuvres to achieve faster lap times.

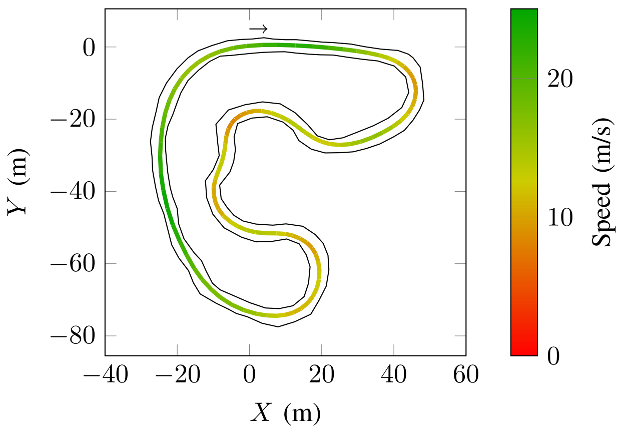

Fig. 9. Simulated trajectory over the second lap of the vehicle in the FSG2019 using LTV-RHP. The beginning of the track and direction the vehicle navigates the track in are indicated by the arrow. Behaviour expected from racing such as corner cutting and slowing down before tight corners can be observed.

C. Computation time

The distribution of computation times measured from all LTV-RHP experiments are shown in Figure 10. For the RHP to run online, it must have a computation time less than the discretised time step $\Delta t = 20\,\mathrm{ms}$. All computation times are observed to be under this threshold. The low median CPU time is also beneficial as it therefore has a negligible contribution to the predicted dead time of 150 ms.

However, a large spread in CPU times was also observed, resulting in a maximum CPU time much greater than the median. This variability was managed by setting the maximum CPU time of the qpOASES solver to 10 ms to constrain the upper bound of computation times. Throughout all of the experiments, this limit was only hit once, representing 0.026% of all solutions. As discussed in Section V, the extremely low probability that this maximum computation time limit is enforced is highly desirable.

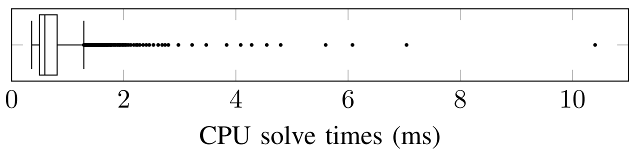

Fig. 10. Distribution of CPU times of LTV-RHP throughout simulated experiments on 10 laps of FSG2019, FSS2019 and FSO2020. Total computation time outliers make up 4.5% of all samples.

VII. Conclusion

This paper presents an LTV-RHP implementation for an autonomous Formula Student racecar to compete in the FSD competition. The use of a successive linearisation discretisation scheme utilising a dynamic bicycle model is proposed to allow the RHP to operate the vehicle closer to the limits of its handling capabilities, while remaining computationally light enough to be feasibly solved in real time. This decision was justified through preliminary experiments which compared it against alternative modelling and discretisation techniques.

The LTV-RHP was validated and compared against MMS's existing motion planners in a hardware-in-the-loop simulation environment, and is shown to successfully outperform MMS's previous motion planning implementation. The improved lap time performance is attributed to an improved vehicle model, improved path tracking capabilities, and improved time delay compensation techniques.

For future work, real-world testing of the LTV-RHP is a natural progression of the experimental work to properly validate and evaluate its on-track performance. Further challenges expected in real-world testing include additional noise, uncertainty, disturbances and time delay which will hinder the performance and stability of the planner.

A second avenue for future work is to modify the RHP for MMS's new electric-driverless vehicle, M21. While M19-D is a rear-wheel drive, M21 is a four-wheel drive with torque vectoring capabilities. Adding an additional control variable in the RHP plant model to account for the torque vectoring capabilities is expected to allow for greater control over the vehicle's behaviour, and further improve the lap times that can be achieved.

References

[1] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[2] G. Hartmann, Z. Shiller, and A. Azaria, "Autonomous head-to-head racing in the Indy autonomous challenge simulation race," *arXiv preprint arXiv:2109.05455*, 2021.

[3] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 1094–1099.

[4] A. Katriniok and D. Abel, "LTV-MPC approach for lateral vehicle guidance by front steering at the limits of vehicle dynamics," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 6828–6833.

[5] B. Alrifaee and J. Maczijewski, "Real-time trajectory optimization for autonomous vehicle racing using sequential linearization," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 476–483.

[6] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit—an open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[7] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle system dynamics*, vol. 21, no. S1, pp. 1–18, 1992.

[8] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[9] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[10] A. Akpunar and S. Iplikci, "Runge-kutta model predictive speed control for permanent magnet synchronous motors," *Energies*, vol. 13, no. 5, p. 1216, 2020.

[11] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[12] C. Smith, *Tune to win*. Aero Publishers Fallbrook, 1978.

[13] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[14] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear model predictive control*. Springer, 2009, pp. 391–417.