A Holistic Motion Planning and Control Solution to Challenge a Professional Racecar Driver

Sirish Srinivasan*,1, Sebastian Nicolas Giles*,1, Alexander Liniger[2]

***Abstract*— We present a holistically designed three layer control architecture capable of outperforming a professional driver racing the same car. Our approach focuses on the co-design of the motion planning and control layers, extracting the full potential of the connected system. First, a high-level planner computes an optimal trajectory around the track, then in real-time the mid-level nonlinear model predictive controller follows this path using the high-level information as guidance. Finally a high frequency, low-level controller tracks the states predicted by the mid-level controller. Tracking the predicted behavior has two advantages: it reduces the mismatch between the model used in the upper layers and the real car, and allows for a torque vectoring command to be optimized by the higher level motion planners. The tailored design of the low-level controller proved to be crucial for bridging the gap between planning and control, unlocking unseen performance in autonomous racing. The proposed approach was verified on a full size racecar, resulting in a considerable improvement over the state-of-the-art results achieved on the same vehicle. Finally, we also show that the proposed co-design approach outperforms a professional racecar driver.**

I. Introduction

Autonomous driving has progressed massively over past few decades, from its humble beginnings in the 1980s [1], [2], over the DARPA challenges [3], [4], to the self-driving car companies of today. One goal for autonomous driving has always been to surpass human driving capabilities, this is especially true for autonomous racing, where professional racecar drivers are a challenging benchmark. However, most existing methods fall short of this goal [5]. Over the last years, several motion planning methods for autonomous racing have emerged [6], [7]. In this paper, we introduce a holistic view-point on motion planning and control of autonomous racecars, and show that the co-design of all layers from track-level trajectory planning to low-level control of the vehicle dynamics allows to achieve unseen performance on a full-sized autonomous racecar. In fact our proposed approach achieves higher driving performance as well as lower best lap times than a professional racing driver, both driving the same Formula Student Driverless (FSD) car developed by AMZ Racing, from ETH Zurich.

Most autonomous racing motion planners and controllers can be divided into three levels. The first level is track-level planning, where the race line around the track is determined. This can be done using either lap-time optimization methods [8]–[10], or using the center line [7], [11], [12]. The mid-level is tasked to follow the track-level path, and is normally based on two common approaches - static feedback controllers [6], [13] or online optimization-based methods [7], [14]–[16] such as Nonlinear Model Predictive Control (NMPC). The last level is the low-level vehicle control, which handles steering and stability control. This layer is often neglected in autonomous racing applications, and solutions developed for human drivers are used [10], [11]. However, in this work, we show that co-designing the low-level controller to work in harmony with the higher levels allows to drastically improve the performance of the autonomous racecar. This reinforces recent work which showed that better coupling the track and mid-level controllers [10], [17] can improve the performance, and [18] that highlighted the benefits of torque vectoring for autonomous cars.

Fig. 1. *pilatus* driverless, the formula student race car ©FSG - Schulz.

A different view point supporting the coupling of different levels in the controller is based on model mismatch. Several papers discovered model mismatch as a crucial issue in autonomous racing - the problem arises due to the relatively simple models used in most autonomous racing stacks. Solutions range from using complex models [17], stochastic MPC [19], to NMPC with model learning [20], [21] to learn the model mismatch. All these methods tackle the problem in the mid-level and come with drawbacks in terms of the computational load. However, using our co-design approach, we can use the low-level controller to make the real-car behave closer to the model of the mid-level NMPC.

In this paper we extend the approach proposed in [10] and highlight the importance of properly coupled high-level and low-level controllers. Our contributions are threefold:

- We propose a new low-level controller designed to distribute the motor torques of our all-wheel drive race car to reduce the model mismatch between the real car and the model used in the higher level layers.
- We propose to directly optimize over a torque vectoring input in the higher level controllers, while interfacing

[1] AMZ Driverless, ETH Zürich, Switzerland
{sirishs,sgiles}@ethz.ch
[2] Computer Vision Lab, ETH Zürich, Switzerland
alex.liniger@vision.ee.ethz.ch
* The authors contributed equally to this work.

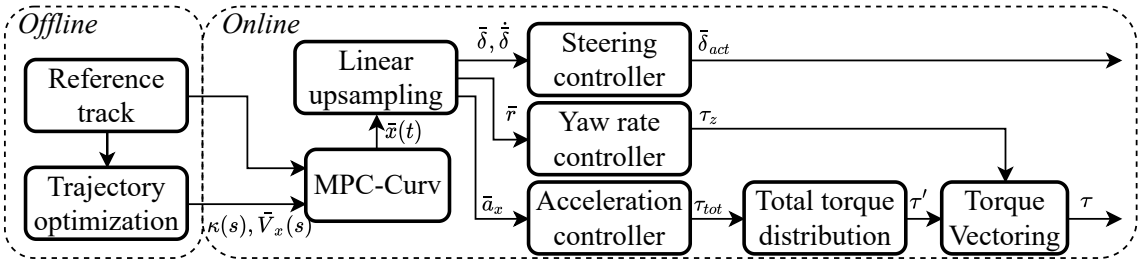

Fig. 2. Full planning and control architecture.

it with the low-level controller in terms of a yaw-rate target trajectory. This allows to extract the full potential out of our low-level control.

- We show that the proposed framework can drastically improve the performance over the current state-of-the-art autonomous racing systems, both in simulation and experiments. In fact our method is the first which performs on par with a professional racecar driver, even outperforming the driver in our experimental setup.

II. Co-designed Controller Architecture

As discussed in the introduction, we build upon [10] for our method, but introduce several fundamental changes highlighted in our first two contributions. However, we keep a similar architecture for the track and mid-level layers, which are the focus of [10]. Our full architecture is shown in Figure 2. We assume a reference path from a mapping run and offline compute an optimal trajectory around the track using our Trajectory Optimization (TRO) module. This path is then followed by the MPC-Curv module, using the terminal velocity constraint from the TRO module. However, in contrast to [10], the optimal solution from MPC-Curv is then translated to set point trajectories for the acceleration, yaw rate and steering, which are tracked using our new low-level controller. At the same time, the redeveloped vehicle model makes the motion planning levels aware of the torque vectoring capability of the car and enables optimization over a new yaw moment command. In addition to these two main differences from [10], we also modify the TRO and MPC-Curv modules, resulting in a drastically improved driving performance.

III. Low Level Control Design

Our fully autonomous racecar (see Section VI-A for full specifications) is equipped with a four wheel independent drive system, where each wheelhub motor can be controlled individually. This requires a sophisticated low-level controller, but also allows significant design freedom. Since the goal is not to simply follow actuator set-points, but to minimize the model mismatch, our low-level controller receives trajectories of vehicle dynamic targets as inputs, namely the longitudinal acceleration $\bar{a}_x$, yaw rate $\bar{r}$ and steering angle $\bar{\delta}$. Note that a bar on top of a variable denotes a target, while no bar refers to the actual variable. The low-level controller then converts these targets into commands for the individual motors and the steering.

A. Wheel Torque Controller

The low-level controller operates at 200Hz, significantly faster than the 40Hz of the upper level NMPC which sends the target trajectories. This allows for a higher bandwidth, but requires in-between target values, which we generate using linear up-sampling of the MPC target trajectories. We further consider the time delay in the robotic system by using slightly time shifted target points.

The low-level controller computes the torques of the four wheels in a two step approach, first the required yaw moment and total torque are computed. In the second step the individual wheel torques are computed fulfilling these requirements.

The yaw moment is computed using a proportional controller to track the target yaw rate $\tau_z = K_P(\bar{r} - r)$. The total torque demand is computed using a PID-controller for the target longitudinal acceleration as $\tau_{\text{total}} = PID(\bar{a}_x - a_x) + m\bar{a}_x + q$, where the feed forward part is designed such that m approximates the inertia of the vehicle including the drive train and q the effect of drag.

An initial torque distribution is then computed by splitting the total torque τ_{total} equally between the left and right sides of the car. Further, the individual torques are scaled proportional to the normal force F_z on each wheel, resulting in $\tau' = [\tau'_{FL}, \tau'_{FR}, \tau'_{RL}, \tau'_{RR}]$. The torque vectoring algorithm then determines the torque differences $\Delta\tau_F$ and $\Delta\tau_R$, which adjusts the wheel torques to $\tau = \tau' + [\Delta\tau_F, -\Delta\tau_F, \Delta\tau_R, -\Delta\tau_R]$. The torque differences $\Delta\tau_F$ and $\Delta\tau_R$ are computed such that the desired yaw moment τ_z is produced, accounting for the effect of the drive force of the angled front wheels. Since the torque difference neglects the load distribution, in a final step, $\Delta\tau_F$ and $\Delta\tau_R$ are distributed proportional to the vertical tire load on each axle. This allows for example to use mainly the rear wheels for torque vectoring during a corner exit.

The resulting wheel torques are finally tracked by a drive motor controller operating at 1kHz. The drive motor controller is also implementing a traction controller, which adapts the reference torque if a slip ratio based wheel speed range is violated [22].

B. Steering Controller

Our steering system is setup as a position servo controller. However, to avoid the delay introduced by the mechanical compliance, we use a virtual target point as a reference. The virtual target point estimates the wheel steering angle given an external steering shaft sensor measurement δ and is given by $\bar{\delta}_{\text{act}} = \bar{\delta} + K_P(\bar{\delta} - \delta) + K_D(\dot{\bar{\delta}} - \dot{\delta})$, where $\dot{\bar{\delta}}(t)$ is the steering rate from the MPC predictions and $\bar{\delta}_{\text{act}}$ is the target for the steering positioning actuator.

IV. Model

Given the low-level controller, we introduce the vehicle model used in the higher level motion planners. Similar to [10] we formulate a dynamic bicycle model in curvilinear coordinates. However, we use control inputs that allow a better interface with our low-level controller.

A. Curvilinear Dynamic Bicycle Model

Curvilinear coordinates describe a coordinate frame (Frenet frame) formulated locally with respect to a reference path, and using these coordinates drastically simplifies path following formulations. In our case the reference path can be the center line or the track-level optimized path. The kinematic states in the curvilinear setting describe the state relative to the reference path and are the progress along the path s, the deviation orthogonal to the path n, and the local heading μ. Note that the dynamic states are not influenced by the change in the coordinate system, and in our model we consider the longitudinal v_x and lateral velocities v_y, and yaw rate r. A visualization of the curvilinear coordinates as well as the other states is shown in Figure 3.

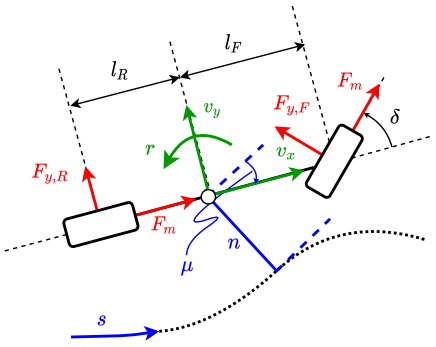

Fig. 3. Visualization of the curvilinear coordinates (blue), as well as the dynamic states (green) and the forces (red).

Since the low-level controller handles traction control considering load changes, we can use a relatively simple model in our higher levels. Our paper follows the popular modeling approach successfully used in [7], [10], [11], and uses a dynamic bicycle model with Pacejka tire models. We follow [10] to formulate the model, but include some important differences - first we assume that the force of the motors F_M can be directly controlled, and for simplicity assume that the same motor force is applied at the front and rear wheels. This approach is better aligned with our low-level controller, which is designed to track an acceleration target and does not expect a driver command as in [10]. Second, we introduce the torque vectoring moment M_{tv} as an input. This is in contrast to [10], [11] where the torque vectoring was determined by a simple P-controller. This input is fundamental, as it allows the higher level controllers to fully utilize the torque vectoring capabilities, going beyond a simple rule-based method designed for human drivers. We use a state lifting technique to consider input rates, but do not lift M_{tv} to allow the high level controllers to use the torque vectoring for highly transient situations. The vehicle state is given by $\tilde{\mathbf{x}} = [s, n, \mu, v_x, v_y, r, F_M, \delta]^T$ and the input as $\mathbf{u} = [\Delta F_M, \Delta\delta, M_{tv}]^T$. Resulting in the following system dynamics,

$$
\begin{aligned}
\dot{s} &= \frac{v_x \cos\mu - v_y \sin\mu}{1 - n\kappa(s)}\,, \\
\dot{n} &= v_x \sin\mu + v_y \cos\mu\,, \\
\dot{\mu} &= r - \kappa(s)\dot{s}\,, \\
\dot{v}_x &= \tfrac{1}{m}(F_M(1+\cos\delta) - F_{y,F}\sin\delta + m v_y r - F_{\text{fric}})\,, \\
\dot{v}_y &= \tfrac{1}{m}(F_{y,R} + F_M\sin\delta + F_{y,F}\cos\delta - m v_x r)\,, \\
\dot{r} &= \tfrac{1}{I_z}((F_M\sin\delta + F_{y,F}\cos\delta) l_F - F_{y,R} l_R + M_{tv})\,, \\
\dot{F}_M &= \Delta F_M\,, \\
\dot{\delta} &= \Delta\delta\,,
\end{aligned}
\tag{1}
$$

where l_F and l_R are the distances from the Center of Gravity (CoG) to the front and rear wheels respectively, m is the mass of the vehicle and I_z the moment of inertia. Finally, $\kappa(s)$ is the curvature of the reference path at the progress s. We denote the dynamics in (1) as $\dot{\tilde{\mathbf{x}}} = f_t^c(\tilde{\mathbf{x}}, \mathbf{u})$, where the superscript c highlights that it is a continuous model and the subscript t that it is a time-domain model.

The lateral forces at the front $F_{y,F}$ and rear $F_{y,R}$ tires are modeled using a simplified Pacejka tire model [23],

$$
\begin{aligned}
F_{y,F} &= F_{N,F} D \sin\left(C \arctan\left(B\alpha_F\right)\right), \\
F_{y,R} &= F_{N,R} D \sin\left(C \arctan\left(B\alpha_R\right)\right),
\end{aligned}
\tag{2}
$$

where $\alpha_F = \arctan\left(\frac{v_y + l_F r}{v_x}\right) - \delta$ and $\alpha_R = \arctan\left(\frac{v_y - l_R r}{v_x}\right)$ are the slip angles at the front and rear wheels respectively, and B, C and D are the parameters of the simplified Pacejka tire model. The net normal load $F_{N,net} = mg + C_l v_x^2$, where C_l is a lumped lift coefficient. Compared to [10] we also consider the aerodynamic downforce, which is important since we push the car to the limit of friction. The resulting normal loads on the front and rear tires are given by $F_{N,F} = F_{N,net} l_R/(l_F + l_R)$ and $F_{N,R} = F_{N,net} l_F/(l_F + l_R)$. Finally, the friction force F_{fric} is a combination of a static rolling resistance C_r and the aerodynamic drag term $C_d v_x^2$.

B. Constraints

Similar to [10] we impose constraints to ensure that the car remains within the track, and that we do not demand inputs that violate friction ellipse or input constraints. More precisely we have a track constraint $\tilde{\mathbf{x}} \in \mathcal{X}_{\text{Track}}$, which is a heading-dependent constraint on the lateral deviation n, and

ensures that the whole car remains inside the track [24],

$$\begin{aligned} n + L_c \sin|\mu| + W_c \cos\mu \leq \mathcal{N}_L(s)\,, \\ -n + L_c \sin|\mu| + W_c \cos\mu \leq \mathcal{N}_R(s)\,, \end{aligned} \tag{3}$$

where L_c and W_c are the distances from the CoG to the furthest apart corner point of the car, and $\mathcal{N}_{R/L}(s)$ are the left and right track width at the progress s. The tire models used (2) do not consider combined slip. So, to prevent the high level layers from demanding unrealistic accelerations from the low-level controller, we limit the combined forces to remain within a friction ellipse,

$$(\rho_{long} F_M)^2 + F_{y,F/R}^2 \leq (\lambda D_{F/R})^2\,. \tag{4}$$

where ρ_{long} defines the shape of the ellipse, and λ determines the maximum combined force. We denote the friction ellipse constraints in (4) by $\tilde{\mathbf{x}} \in \mathcal{X}_{\text{FE}}$.

Finally, we consider box constraints for both the physical inputs and their rates. We introduce a compact notation for all these inputs $\mathbf{a} = [F_M, \delta, \Delta F_M, \Delta\delta, M_{tv}]^T$, and constrain them to their physical limits by the box constraint $\mathbf{a} \in \mathcal{A}$.

V. High Level Control Formulation

A. Trajectory Optimization (TRO)

Given the vehicle model and the constraints, we now focus on the higher level controllers. First the offline track-level trajectory optimization is described. Following [10], we transform the continuous time dynamics (1) into the spatial domain, with progress s as the running variable instead of time t. This transformation can be achieved as follows,

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \frac{\partial \tilde{\mathbf{x}}}{\partial t} = \frac{\partial \tilde{\mathbf{x}}}{\partial s}\frac{\partial s}{\partial t}\,, \\ \Rightarrow \frac{\partial \tilde{\mathbf{x}}}{\partial s} &= \frac{1}{\dot{s}} f_t^c(\tilde{\mathbf{x}}(s), \mathbf{u}(s)) = f_s^c(\tilde{\mathbf{x}}(s), \mathbf{u}(s))\,. \end{aligned} \tag{5}$$

where $f_s^c(\tilde{\mathbf{x}}(s), \mathbf{u}(s))$ is the continuous space model. This transformation also makes the s state redundant allowing us to reduce the state to $\mathbf{x} = [n, \mu, v_x, v_y, r, F_M, \delta]^T$.

To formulate the track-level optimization problem, we use the center line reference path and discretize the continuous space model $f_s^c(\mathbf{x}(s), \mathbf{u}(s))$. An Euler forward integrator, with a discretization distance of Δs is used, resulting in the discrete space system $\mathbf{x}_{k+1} = f_s^d(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k + \Delta s f_s^c(x_k, \mathbf{u}_k)$. For racing, it is necessary to optimize for a periodic trajectory. Thus, we add the periodicity constraint to our optimization problem $f_s^d(\mathbf{x}_N, \mathbf{u}_N) = \mathbf{x}_0$, where N is the number of discretization steps.

The cost function seeks to maximize the progress rate $\dot{s}$, and also contains two regularization terms - a slip angle cost and a penalty on the input rates. The slip angle cost penalizes the difference between the kinematic and dynamic side slip angles as $B(\mathbf{x}_k) = q_\beta(\beta_{\text{dyn,k}} - \beta_{\text{kin,k}})^2$, where $q_\beta > 0$ is a weight, $\beta_{\text{kin,k}} = \arctan(\delta_k l_R/(l_F + l_R))$, and $\beta_{\text{dyn,k}} = \arctan(v_{y,k}/v_{x,k})$. The regularizer on the input rates is $\mathbf{u}^T R \mathbf{u}$, where R is a diagonal weight matrix with positive weights. In summary, the overall cost function is,

$$J_{\text{TRO}}(\mathbf{x}_k, \mathbf{u}_k) = -\dot{s}_k + \mathbf{u}^T R \mathbf{u} + B(\mathbf{x}_k)\,. \tag{6}$$

Note that we introduced a new cost function compared to [10], which minimized the time. Our new cost function is nearly identical to the one used in the MPC-Curv motion planner (Section V-B), which makes tuning easier, and better aligns the solutions of the two levels.

Finally, we combine the cost, model and model related constraints from Section IV-B, to formulate the trajectory optimization problem,

$$\begin{aligned} \min_{\mathbf{X},\mathbf{U}} \quad & \sum_{k=0}^{N} J_{\text{TRO}}(\mathbf{x}_k, \mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = f_s^d(\mathbf{x}_k, \mathbf{u}_k)\,, \\ & f_s^d(\mathbf{x}_N, \mathbf{u}_N) = \mathbf{x}_0\,, \\ & \mathbf{x}_k \in \mathcal{X}_{\text{Track}}, \quad \mathbf{x}_k \in \mathcal{X}_{\text{FE}}\,, \\ & \mathbf{a}_k \in \mathcal{A}, \quad k = 0, ..., N\,, \end{aligned} \tag{7}$$

where $\mathbf{X} = [\mathbf{x}_0, ..., \mathbf{x}_N]$ and $\mathbf{U} = [\mathbf{u}_0, ..., \mathbf{u}_N]$. The problem is formulated in the modeling package *CppAD* [25] and solved using *Ipopt* [26].

B. MPC-Curv

We use MPC-Curv to follow the trajectory from TRO. Since the objectives of TRO and MPC-Curv are similar, we reuse large parts of the formulation, including the progress rate maximization objective. However, since the MPC-Curv problem is solved online, a few changes are necessary - first, we use the time-domain model since it better suited for online control (see [10] for a discussion). Second, since MPC-Curv has a limited prediction horizon, we use the TRO solution for long term guidance. This is done in two ways - first a regularization cost on the lateral deviation n from the TRO path and more importantly a terminal velocity constraint to notify MPC-Curv about upcoming braking spots beyond the prediction horizon.

To formulate the MPC-Curv problem, we first reduce the state to $\mathbf{x} = [n, \mu, v_x, v_y, r, F_M, \delta]^T$, by relying on the initial guess of s to evaluate all quantities depending on s such as the curvature κ. Since we do not evaluate $\kappa(s)$ inside the MPC, the s-state decouples. Second, we discretize the model using a second order Runge Kutta method, resulting in $f_t^d(\mathbf{x}_t, \mathbf{u}_t)$. In order to decouple the prediction horizon in terms of steps and time, we discretize the system with a sampling time different from the update frequency of the controller. We introduce this as a time scaling factor σ which multiplies the sampling time of our controller. Note that this requires interpolating the previous optimal solution to get an initial guess, but at the same time allows to predict further in time with the same horizon in steps. This allows us to run MPC-Curv with higher update rates.

The MPC-Curv cost function is identical to the TRO (6), with the addition of a path following cost on n,

$$J_{\text{MPC}}(\mathbf{x_t}, \mathbf{u}_t) = -\dot{s}_t + q_n n_t^2 + \mathbf{u}^T R \mathbf{u} + B(\mathbf{x}_t)\,, \tag{8}$$

where q_n is a positive regularization weight. Thus, we can now formulate the MPC-Curv problem,

$$
\begin{aligned}
\min_{\mathbf{X},\mathbf{U}} \quad & \sum_{t=0}^{T} J_{\text{MPC}}(\mathbf{x}_t, \mathbf{u}_t) \\
\text{s.t.} \quad & \mathbf{x}_0 = \hat{\mathbf{x}}\,, \\
& \mathbf{x}_{t+1} = f_t^d(\mathbf{x}_t, \mathbf{u}_t)\,, \\
& \mathbf{x}_t \in \mathcal{X}_{\text{Track}}, \quad \mathbf{x}_t \in \mathcal{X}_{\text{FE}}\,, \\
& \mathbf{a}_t \in \mathcal{A}, \quad v_{x,T} \leq \bar{V}_x(s_T)\,, \\
& t = 0, ..., T\,,
\end{aligned} \tag{9}
$$

where the subscript t is used to highlight that the problem is formulated in the time domain, and T is the prediction horizon. Further, $\hat{\mathbf{x}}$ is the current curvilinear state estimate and $v_{x,T} \leq \bar{V}_x(s_T)$ the terminal constraint from the TRO solution. The optimization problem is solved in real-time using ForcesPro [27], [28], a proprietary interior point method solver optimized for NMPC problems.

VI. Results and Discussion

We first discuss our robotic platform - the autonomous racecar *pilatus driverless*, followed by implementation details. Thereafter, we then benchmark our control approach in simulation using a realistic vehicle dynamics simulator and study the effect of the low-level adaptations. Finally, we present experimental results including an in-depth comparison with a professional racing driver.

A. pilatus driverless

All our experiments are performed using the autonomous racecar *pilatus driverless* (shown in Figure 1). *pilatus* is a lightweight single seater race car, with an all-wheel drive electric powertrain. The racecar can produce $\pm 375\,\text{Nm}$ of torque at each wheel by means of four independent $38.4\,\text{kW}$ motors. Our racecar can accelerate from $0 - 100\,\text{km/h}$ in $2.1\,\text{s}$ and can reach lateral accelerations of over $20\,\text{m/s}^2$. *pilatus* is equipped with a complete sensor suite including two LiDARs, three cameras, an absolute speed sensor and two IMUs. The low-level control as well as the state estimation are deployed on an ETAS ES900 real-time embedded system; the remainder of the Autonomous System (AS), including mapping and localization, runs on an Intel Xeon E3 processor, see [11], [29]–[32] for more details.

B. TRO and MPC-Curv Implementation Details

TRO uses a spatial discretization of $\Delta s = 0.5\,\text{m}$. Given this discretization, the optimization problem defined in (7) is solved in $\sim 5\,\text{s}$ on an Intel Xeon E3 processor. We run MPC-Curv at a frequency of 40Hz, with a time scaling $\sigma = 1.5$. We use a prediction horizon of $T = 40$ which results in a time horizon of $1.5\,\text{s}$, using the time scaling.

C. Simulation Study

To highlight the benefits achieved by co-designing the low-level and the higher level controllers, we compare our full pipeline against a modified version of it that cannot optimize over torque vectoring and uses the more basic low-level controller as in [10]. Note that the second method is similar to [10]. We perform a simulation on the Formula Student Germany racetrack from 2018, using a high fidelity vehicle dynamics simulator. As an upper performance bound we also include the TRO solution, which uses the dynamic bicycle model (1) with no mismatch. Figure 4 shows the longitudinal velocity v_x against the progress along the track. When comparing the two methods simulated on the high fidelity model, it is clearly visible that our approach can reach higher speeds. The lap times of the three methods confirm this: our full system completes one lap in $19.9\,\text{s}$ whilst without the low-level adaptations the lap time is $22.1\,\text{s}$, for reference the TRO optimal lap time is $18.0\,\text{s}$.

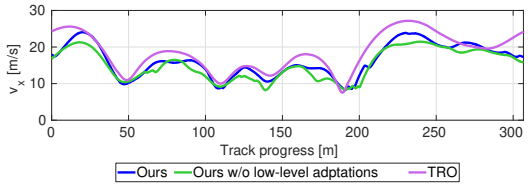

Fig. 4. Longitudinal velocity against track position (in simulation), comparing our approach with and w/o the proposed low-level adaptations, against TRO.

D. Experimental Comparison

For our comparison with a professional racing driver, we setup a race track compliant with the Formula Student Driverless regulations [33], composed of sharp turns, straights and chicanes.

We tried to keep the comparison between the human driver and our proposed solution as fair as possible, however, there are some differences. First, in autonomous mode, the car is lighter due to the absence of a driver. Second, for safety, the speed when driving autonomously was limited to $18\,\text{m/s}$, whereas the human driver was unrestricted and reaches speeds up to $22\,\text{m/s}$. Third, when driving autonomously the car can only use regenerative braking, since the hydraulic brakes are exclusively used for an emergency braking system. Since this is not necessary for the human driver, the hydraulic brakes can also be actuated, allowing higher deceleration. We would also like to note that the steering system used in the autonomous mode is slower than the steering actuated by a driver. The last difference is the low-level controller, where the human uses the "standard" low-level controller, which uses a different torque vectoring module [34], but the same traction controller.

The experiments were run consecutively, with one run for each of the driving modes. The duration was of 12 and 18 laps for the human and AS respectively. In both cases, the car remained within the track and did not hit any cones.

1) Lap-time comparison: All of the lap times recorded during the experiment are shown in Table I and Figure 5, no data was discarded. Our proposed controller achieves both a lower average lap time, as well the lowest lap time. In Figure 5 we can see that the autonomous system achieves six laps that are faster than the fastest lap by the professional human driver. Note that the variability of lap times in

the autonomous mode comes from a few instances where an emergency planner is triggered and automatically slows down the car to avoid leaving the track.

TABLE I

LAP-TIME COMPARISON BETWEEN HUMAN AND AUTONOMOUS DRIVER

	Human	Autonomous
Best lap-time(s)	13.62	13.39
Mean lap-time(s)	14.19	13.95

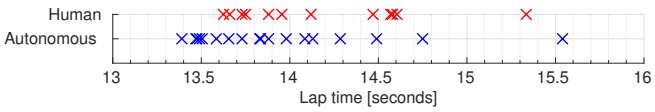

Fig. 5. Lap-time distribution for autonomous system and human driver.

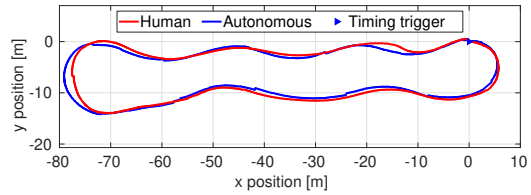

Fig. 6. GPS trajectory comparison of the best autonomous and human driven laps. The driving direction is clockwise.

2) Driving comparison: Figure 6 shows the comparison of the driven GPS paths for the autonomous and human modes, and Figure 7 compares the longitudinal velocity of all driven laps. When comparing the driven paths, one major difference between the paths can be seen at the increasing radius curve on the left extreme. The AS does not follow the intuitive inner radius of the curve but goes wide, which allows later braking and a faster and straighter curve exit, which we can see in Figure 7 at 100 m. A similar difference can also be seen in the curve at the right extreme, where especially the last chicane before the curve entry and the curve exit are driven faster. The offline TRO problem can efficiently perform such trade-offs between travelled distance and speed, while even a professional driver needs significant track time to evaluate such trade-offs.

The other significant difference we can see in Figure 7 is the effect of the 18 m/s speed limiter for the AS. However, our controller is able to brake later and accelerate earlier at several locations along the lap, which in total results in lower lap-times, even with the top speed disadvantage.

3) Inputs comparison: When comparing the low-level inputs in Figure 7, it is interesting to note that the AS appears to make limited use of the steering, which stands in contrast to the human driver. At the same time the torque vectoring system is used significantly more, both in terms of magnitude and frequency. This shows that our holistic architecture exploits the improved torque vectoring to achieve a faster transient behavior.

The final difference is the relative total torque input, where we can see that the AS demands less torque. This is due to

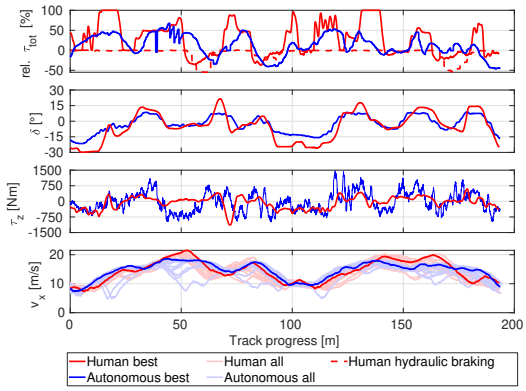

Fig. 7. Comparison of the vehicle dynamics on the best lap. The velocity in all of the laps is also shown in a lighter color.

the speed limit, and the fact that the friction ellipse constraint limits the torque before the traction controller. The human driver on the other hand relies on the traction controller in certain situations, e.g., at 100 m.

4) Maximum performance: Figure 8 compares the longitudinal and lateral accelerations recorded by the car in autonomous and human driven modes. We can see that the highest lateral, positive longitudinal and combined accelerations are achieved by the autonomous controller. The human driver has a higher negative longitudinal acceleration, due to the availability of hydraulic brakes, which are not used by the autonomous controller. Finally, we can also see the drastic performance difference between our approach and [10], while using the same car.

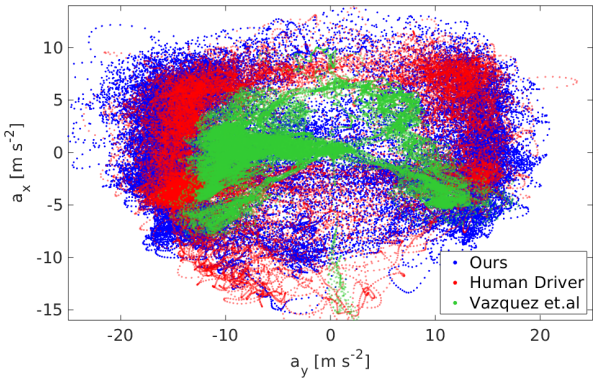

Fig. 8. Longitudinal and lateral acceleration comparison between our system, the professional human driver and Vazquez et.al [10]

VII. CONCLUSION

In this paper we proposed a holistic way to think about a motion planner and controller design for autonomous racing. The idea is that all hierarchical control layers should be designed while keeping the other layers in mind. We proposed a low-level controller that actuates the steering, and distributes the wheel torques to track the acceleration, yaw rate and steering trajectories predicted by a higher level NMPC. The higher level motion planners, again consider

the torque vectoring capabilities of the low-level controller. Thus, the model mismatch between the levels can be reduced, while the capabilities of the car can be fully extracted. We show this by comparing the performance of our autonomous controller with a professional human driver both driving the same full-sized autonomous racecar. Our autonomous controller is able to better the driver both in peak and average lap-times. Future work will include real-time identification of the peak tire performance to benefit from the full grip potential, and data-driven learning of the cost function.

ACKNOWLEDGMENT

We would like to thank the entire AMZ Driverless team, this work would not have been possible without the effort of every single member, and we are glad for having the opportunity to work with such amazing people. We would also like to thank the numerous alumni for the insightful discussions.

REFERENCES

[1] E. D. Dickmanns and A. Zapp, "Autonomous high speed road vehicle guidance by computer vision," *IFAC Proceedings Volumes*, vol. 20, no. 5, pp. 221–226, 1987.

[2] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, 1989.

[3] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*. Springer, 2007.

[4] ——, *The DARPA urban challenge: autonomous vehicles in city traffic*. Springer, 2009.

[5] L. Hermansdorfer, J. Betz, and M. Lienkamp, "Benchmarking of a software stack for autonomous racing against a professional human race driver," in *Ecological Vehicles and Renewable Energies (EVER)*, 2020.

[6] K. Kritayakirana and J. C. Gerdes, "Autonomous vehicle control at the limits of handling," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 4, pp. 271–296, 2012.

[7] 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.

[8] R. Lot and F. Biral, "A curvilinear abscissa approach for the lap time optimization of racing vehicles," *IFAC World Congress*, 2014.

[9] A. Rucco, G. Notarstefano, and J. Hauser, "An efficient minimum-time trajectory generation strategy for two-track car vehicles," *Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1505–1519, July 2015.

[10] J. L. Vazquez, M. Bruhlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," in *International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[11] J. Kabzan, M. I. Valls, V. J. F. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart, "AMZ driverless: The full autonomous racing system," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020.

[12] U. Rosolia, A. Carvalho, and F. Borrelli, "Autonomous racing using learning model predictive control," in *American Control Conference (ACC)*, 2017.

[13] J. Betz, A. Wischnewski, A. Heilmeier, F. Nobis, L. Hermansdorfer, T. Stahl, T. Herrmann, and M. Lienkamp, "A software architecture for the dynamic path planning of an autonomous racecar at the limits of handling," in *International Conference on Connected Vehicles and Expo (ICCVE)*, 2019.

[14] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, "Collision avoidance and stabilization for autonomous vehicles in emergency scenarios," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1204–1216, 2016.

[15] D. Caporale, A. Settimi, F. Massa, F. Amerotti, A. Corti, A. Fagiolini, M. Guiggian, A. Bicchi, and L. Pallottino, "Towards the design of robotic drivers for full-scale self-driving racing cars," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[16] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," *International Conference on Robotics and Automation (ICRA)*, 2016.

[17] T. Novi, A. Liniger, R. Capitani, and C. Annicchiarico, "Real-time control for at-limit handling driving on a predefined path," *Vehicle System Dynamics*, vol. 58, no. 7, pp. 1007–1036, 2020.

[18] C. Chatzikomis, A. Sorniotti, P. Gruber, M. Zanchetta, D. Willans, and B. Balcombe, "Comparison of path tracking and torque-vectoring controllers for autonomous electric vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 4, pp. 559–570, 2018.

[19] J. V. Carrau, A. Liniger, X. Zhang, and J. Lygeros, "Efficient implementation of randomized MPC for miniature race cars," *European Control Conference (ECC)*, 2016.

[20] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[21] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli, "Repetitive learning model predictive control: An autonomous racing example," *Conference on Decision and Control (CDC)*, 2017.

[22] D. Bohl, N. Kariotoglou, A. B. Hempel, P. J. Goulart, and J. Lygeros, "Model-based current limiting for traction control of an electric four-wheel drive race car," in *European Control Conference (ECC)*, 2014.

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

[24] A. Liniger and L. V. Gool, "Safe motion planning for autonomous driving using an adversarial road model," in *Robotics: Science and Systems (RSS)*, 2020.

[25] B. Bell. (2019) Cppad: A package for c++ algorithmic differentiation. [Online]. Available: http://www.coin-or.org/CppAD

[26] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[27] A. Domahidi and J. Jerez, "Forces professional," Embotech AG", url="https://embotech.com/FORCES-Pro, 2014–2021.

[28] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, 2020.

[29] M. I. Valls, H. F. Hendrikx, V. J. Reijgwart, F. V. Meier, I. Sa, R. Dubé, A. Gawel, M. Bürki, and R. Siegwart, "Design of an autonomous racecar: Perception, state estimation and system integration," in *International Conference on Robotics and Automation (ICRA)*, 2018.

[30] N. Gosala, A. Bühler, M. Prajapat, C. Ehmke, M. Gupta, R. Sivanesan, A. Gawel, M. Pfeiffer, M. Bürki, I. Sa *et al.*, "Redundant perception and state estimation for reliable autonomous racing," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[31] S. Srinivasan, I. Sa, A. Zyner, V. Reijgwart, M. I. Valls, and R. Siegwart, "End-to-end velocity estimation for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6869–6875, 2020.

[32] L. Andresen, A. Brandemuehl, A. Honger, B. Kuan, N. Vödisch, H. Blum, V. Reijgwart, L. Bernreiter, L. Schaupp, J. J. Chung *et al.*, "Accurate mapping and planning for autonomous racing," in *International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[33] F. S. G. R. 2019. Fs rules 2019 v1.1.

[34] W. F. Milliken and D. L. Milliken, *Race Car Vehicle Dynamics*. Great Britain: Society of Automotive Engineers Inc., 1996.