

# Model Parameter Identification via a Hyperparameter Optimization Scheme for Autonomous Racing Systems

Hyunki Seong<sup>1</sup>, Chanyoung Chung<sup>2\*</sup>, and David Hyunchul Shim<sup>1</sup>

**Abstract**—In this letter, we propose a model parameter identification method via a hyperparameter optimization scheme (MI-HPO). Our method adopts an efficient explore-exploit strategy to identify the parameters of dynamic models in a data-driven optimization manner. We utilize our method for model parameter identification of the AV-21, a full-scaled autonomous race vehicle. We then incorporate the optimized parameters for the design of model-based planning and control systems of our platform. In experiments, MI-HPO exhibits more than 13 times faster convergence than traditional parameter identification methods. Furthermore, the parametric models learned via MI-HPO demonstrate good fitness to the given datasets and show generalization ability in unseen dynamic scenarios. We further conduct extensive field tests to validate our model-based system, demonstrating stable obstacle avoidance and high-speed driving up to 217 km/h at the Indianapolis Motor Speedway and Las Vegas Motor Speedway. The source code for our work and videos of the tests are available at <https://github.com/hynkis/MI-HPO>.

**Index Terms**—Data-driven control, hyperparameter optimization, autonomous vehicle

## I. INTRODUCTION

UNDERSTANDING the system model is essential for robotic applications, especially safety-critical autonomous systems. In particular, at high-speed driving, various dynamics elements such as chassis, tires, or engines become crucial to implement high-speed autonomy. Model-based optimal control [1], [2] is well-suited for handling those factors and is widely used to design dynamics system control. By leveraging physics-based parametric dynamic models, it optimizes driving behavior with respect to a designed objective function and enables safe and reliable control system design.

Despite the success of the model-based approach in robotics, model-based algorithms have two fundamental challenges: model fidelity and tractability. The performance of model-based approaches relies heavily on the accuracy of the model.

However, identifying accurate models is often laborious or intractable because of their large search space and nonlinearity. Other than the model accuracy, models also need to be computationally feasible for real-time control applications. High-fidelity but highly complex models are often difficult to integrate into real-time safety-critical driving systems.

To tackle these challenges, conventional approaches, including the Prediction Error Method, have been used to identify model parameters [3]. However, those methods often require the model structure to be linear or in specific mathematical forms, which might not be feasible to describe the nonlinear high-speed autonomous driving system. Gradient-based optimization (GBO) [4] methods are often employed for parameter identification by updating parameters with a function's gradient vector. However, GBO methods can be sensitive to the initial guess and learning rate of parameter updates. They also necessitate a differentiable model and objective function. On the other hand, another traditional approach, particle swarm optimization (PSO) [5] is a stochastic optimization that adjusts candidate solutions iteratively without requiring differentiability. Nevertheless, it can be highly affected by the choice of its parameters, such as the number of particles and acceleration coefficients. Poorly chosen parameter values can lead to slow or premature convergence to suboptimal solutions. Recent studies have utilized neural networks (NNs) to model nonlinear dynamics, such as using a simple NN to replace a single-track model [6] or an NN-based model approximator to identify the vehicle dynamics model in an end-to-end learning fashion [7]. However, integrating NNs with non-learning model-based reliable methods in real-world applications can be difficult. Additionally, ensuring the validity of the NN model in unseen driving scenarios without large-scale field tests is challenging.

In this letter, we propose a model parameter identification method via a hyperparameter optimization scheme (MI-HPO) to efficiently optimize the configuration of model parameters for high-speed autonomous racing systems. Hyperparameter optimization (HPO) techniques are used to select the optimal hyperparameter configuration for a learning algorithm in machine learning. As the learning process can take a substantial amount of time, these techniques prioritize a balanced exploration and exploitation strategy to efficiently select hyperparameters [8]. Our key idea is to leverage an HPO approach to identify physics-based parametric models in a data-driven manner without any limitation on the form

This research was financially supported by the Institute of Civil Military Technology Cooperation funded by the Defense Acquisition Program Administration and Ministry of Trade, Industry and Energy of Korean government under grant No. UM22206RD2.

\*Corresponding author

<sup>1</sup> The authors are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea. (email: hynkis@kaist.ac.kr; geninfy@kaist.ac.kr)

<sup>2</sup> Chanyoung Chung is with the JPL Science Division, NASA, California, USA. (email: chanyoung.chung@jpl.nasa.gov)

of the model equation. To this end, we adopt Hyperband [9], a novel HPO scheme with an explore-exploit strategy, while incorporating mutation and annealing methods to design the MI-HPO algorithm. Using our method, we successfully estimate the parameters of the integrable parametric dynamics models for a full-scaled autonomous racecar, Dallara AV-21 (Fig. 1), at the Indy Autonomous Challenge (IAC) [10]. We further investigate the advantages of MI-HPO by comparing its performance with conventional optimization methods. Finally, We validate our method by integrating the identified models into the high-speed autonomous system and conducting extensive field experiments, including over 200 km/h driving and obstacle avoidance scenarios in the Indianapolis Motor Speedway (IMS) and Las Vegas Motor Speedway (LVMS).

The rest of the letter is organized as follows. Sec. II presents the MI-HPO algorithm; Sec. III and IV introduce vehicle dynamics models and model-based planning/control for our racing system, respectively; Sec. V contains our main evaluation results; and finally, Sec. VI concludes the letter.

## II. MODEL PARAMETER IDENTIFICATION

In this section, we formulate the model parameter identification problem combined with an HPO approach. First, we regard a parameter configuration  $p \in \mathbb{R}^{n_p}$  with  $n_p$  model parameters as a set of hyperparameters of a nonlinear dynamics model  $f$ . Then, we identify the parameter configuration by evaluating the following objective function inspired by the standard supervised learning problem:

$$\mathcal{L} = \frac{1}{|D|} \sum_{(x,y) \in D} (y - f(x; p))^2, \quad (1)$$

where  $x, y$  denote the sampled input and output data of the model  $f$  from a given dataset  $D$ . By minimizing this learning objective, we find an optimized configuration  $p^*$  that has the minimum model error with the observed model output  $y$ . The model  $f$  has no limitation on its form of the equation. Thus, our method can be used for arbitrary parametric models.

We implement MI-HPO incorporating a bandit-based algorithm, Hyperband [9], as summarized in Algorithm 1. It is a variation of a random search algorithm with exploitation-exploration features to find the optimal configuration based on an evaluation loss. The algorithm requires two arguments:  $R$ , the maximum amount of resource (e.g., the number of

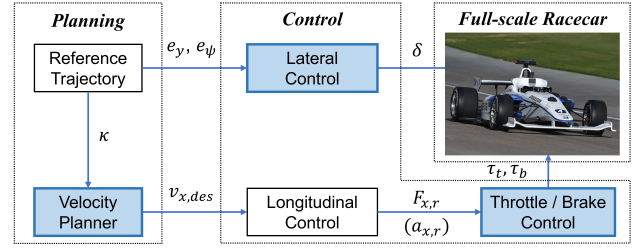


Fig. 1. Overview of our autonomous driving system in the AV-21. Our learned model parameters are embedded in the planning and control modules that are covered in this letter (highlighted in blue).

evaluation iterations) that can be allocated to a single configuration, and  $\eta$ , a value that determines the proportion of the discarded configurations. The two arguments initially set  $s_{max}$  and a budget  $B$ , which derive  $s_{max} + 1$  combinations of the values  $n$  and  $r$ . They enable various ratios of exploration and exploitation to find the optimal configuration. The algorithm consists of two loops. In the outer loop, as  $s$  is gradually decreased, the number of samples  $n$  is reduced, and conversely, the number of evaluation resources  $r$  is increased. This encourages exploration with more samples in the early optimization stages and enhances exploitation by performing more evaluations with fewer samples as the iteration progresses, allowing for efficient optimization. In the inner loop, the algorithm compares the loss of each sample and allocates more resources to the samples with lower evaluation losses, excluding the ones with higher losses. It repeats the sampling and exclusion processes until the last configuration remains.

To adjust the original HPO process to the model parameter optimization, we apply an additional Gaussian mutation process [11] in the evaluation process to explore new neighboring parameters with lower model loss. Unlike the original HPO algorithm, which only allocates more resources  $r_i$  without modifying initial configurations, our approach adds perturbation with a noise random variable  $\epsilon \in \mathbb{R}^{n_p}$  at the selected configuration  $p$  after the resource allocation as:

$$p_{mut} = p + \sigma \odot \epsilon, \quad \epsilon \sim N(0, I), \quad (2)$$

where  $\odot$  is the element-wise product and  $\sigma \in \mathbb{R}^{n_p}$  is the standard deviation of the exploration noise. To achieve fine parameter updates, we linearly anneal  $\sigma$  from its maximum value to its minimum value over the course of the evaluation. We define the following three functions for MI-HPO:

- *get\_model\_param\_config*( $n$ ): a function that returns a set of  $n$  random parameter configurations from the normal distribution pre-defined over the configuration space.
- *eval\_with\_mutation*( $p, r_j, D$ ): a function that receives a parameter configuration  $p$ , an allocated resource  $r_j$ , and a dataset  $D$  as arguments. For a total of  $r_j$  iterations, the function repeatedly mutates the initial configuration according to Eq. 2 and evaluates it using Eq. 1 and  $D$ . If a mutated configuration  $p_{mut} \in \mathbb{R}^{n_p}$  has a less loss than the initial one, the function replaces  $p$  with  $p_{mut}$ . It returns the final loss after spending the allocated iterations.
- *select\_top\_k\_config*( $P, L, k$ ): a function that receives a set of hyperparameter configurations  $P$  with their corresponding evaluation losses  $L$  and returns the top  $k$  high-performing configurations (here,  $k = \lfloor n_j / \eta \rfloor$ ).

---

### Algorithm 1 MI-HPO Algorithm based on Hyperband

---

**Input:**  $R, \eta, D$

- 1:  $s_{max} \leftarrow \lfloor \log_{\eta}(R) \rfloor, B = (s_{max} + 1)R$
- 2: **for**  $s \in \{s_{max}, s_{max} - 1, \dots, 0\}$  **do**
- 3:    $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil, r = R\eta^{-s}$
- 4:    $P = \text{get\_model\_param\_config}(n)$
- 5:   **for**  $j \in \{0, \dots, s\}$  **do**
- 6:      $n_j = \lfloor n\eta^{-j} \rfloor, r_j = r\eta^j$
- 7:      $L = \{\text{eval\_with\_mutation}(p, r_j, D) : p \in P\}$
- 8:      $P = \text{select\_top\_k\_config}(P, L, \lfloor n_j / \eta \rfloor)$

**Output:** Optimized parameters  $p^*$  with the smallest loss.

---

### III. VEHICLE DYNAMICS MODEL

#### A. Tire Dynamics Model

The tire model is one of the factors that significantly affect the nonlinearity of driving dynamics. Especially the lateral tire model is crucial to design stable path-tracking control in high-speed driving. The tire model [12] can be described as a function of the slip angle  $\alpha_i$ , slip ratio  $\rho_{x,i}$ , inclination angle  $\theta_i$ , tire load  $F_{z,i}$ , and current velocity  $v_{x,i}$ , which has a lateral tire force  $F_{y,i}^*$  of each tire ( $i \in \{LF, LR, RF, RR\}$ ) as,

$$F_{y,i}^* = f_{tire}(\alpha_i, \rho_{x,i}, \theta_i, F_{z,i}, v_{x,i}). \quad (3)$$

Although the model has high fidelity with various dynamics perspectives, it has low suitability for designing the controller of high-speed driving, which requires real-time performance. Therefore, we first define a tire model with dimension-reduction that can be applied to model-based control design within an acceptable complexity. We then optimize the model's parameter configuration to represent the overall tire characteristic of a given dataset using our MI-HPO algorithm. We follow the Pacejka tire model [13] to define the tire dynamics. While the prior work neglect vertical and horizontal offsets of the model, we formulate a tire model  $F_{y,i} = f_{t,i}(\alpha_i; p_{t,i})$  containing the offset parameters  $S_{x,i}, S_{y,i}$  to describe the asymmetric tire characteristic determined to maximize cornering performance on an oval track:

$$F_{y,i} = D_i \sin(C_i \arctan(B_i(\alpha_i + S_{x,i}))) + S_{y,i}, \quad (4)$$

where the tire model parameter configuration  $p_{t,i} = \{B_i, C_i, D_i, S_{x,i}, S_{y,i}\}$  is identified by minimizing the following tire model objective with a given dataset  $D_{t,i}$  as

$$\mathcal{L}_{t,i} = \frac{1}{|D_{t,i}|} \sum_{(\alpha_i, F_{y,i}^*) \in D_{t,i}} (F_{y,i}^* - f_{t,i}(\alpha_i; p_{t,i}))^2. \quad (5)$$

#### B. Engine Torque Model

The powertrain system of our racecar consists of an internal combustion engine, transmission, and wheels. The AV-21 is a rear-wheel-drive vehicle whose traction force  $F_{x,r}$  is generated by engine-based driveline dynamics. We model the equation of the longitudinal dynamics [14] as follows:

$$ma_x = F_{x,r} - C_d v_x^2 - C_r, \quad (6)$$

where  $m$  is the vehicle mass,  $v_x$  is the longitudinal velocity, and  $C_d, C_r$  denote the drag and rolling coefficient, respectively<sup>1</sup>. Following a prior work [15], the traction force can be expressed as:

$$F_{x,r} = ma_{x,r} = \frac{T_e \eta_t i_g i_0}{R_w}, \quad (7)$$

where  $a_{x,r}$  denotes the traction acceleration,  $\eta_t$  denotes the efficiency of the transmission,  $i_g, i_0$  denote the transmission ratio of the current gear and final reducer, and  $R_w$  denotes the wheel radius.  $T_e = f_e(w_e, \tau_t)$  is the engine torque map in terms of the engine speed  $w_e$  and throttle command  $\tau_t$ . Due to the high complexity of the engine characteristic, the

torque map is expressed as an experimental lookup table based on engine torque curves of specific throttle opening commands. Although the engine torque model can be obtained by the engine dynamometer testing [16], it could suffer from modeling errors because the dynamometer testing is done in a static environment. Therefore, we build the engine torque map that integrates the dyno data<sup>2</sup> with learned torque curves based on our data-driven model identification approach. We express an engine torque curve  $T_{e,\tau_t} = f_{\tau_t}(w_e; p_{\tau_t})$  of a throttle command  $\tau_t$  as a third order polynomial function of the engine speed  $w_e$  as:

$$T_{e,\tau_t} = p_{\tau_t,0} + p_{\tau_t,1}w_e + p_{\tau_t,2}w_e^2 + p_{\tau_t,3}w_e^3, \quad (8)$$

where  $p_{\tau_t} = \{p_{\tau_t,0}, p_{\tau_t,1}, p_{\tau_t,2}, p_{\tau_t,3}\}$  is the torque model parameter configuration. Using the resultant traction accelerations  $a_{x,r}$  that can be derived by Eq. 6 while driving, the engine torque output  $T_{e,\tau_t}$  is obtained by Eq. 7. Then,  $p_{\tau_t}$  is learned by minimizing the following engine model objective with a given dataset  $D_{\tau_t}$ :

$$\mathcal{L}_{\tau_t} = \frac{1}{|D_{\tau_t}|} \sum_{(w_e, T_{e,\tau_t}^*) \in D_{\tau_t}} (T_{e,\tau_t}^* - f_{\tau_t}(w_e; p_{\tau_t}))^2. \quad (9)$$

To stabilize the learning process, we normalize the engine speed in the range [0,1] with the maximum engine speed.

### IV. MODEL-BASED PLANNING AND CONTROL

In this session, we introduce a model-based planning and control algorithm that uses the learned model parameters. We exploit the learned tire parameters to design a dynamics-aware velocity planning and model-based lateral controller (Fig. 1). We also integrate the engine dyno data with the learned engine torque model to construct an engine lookup table.

#### A. Dynamics-aware Velocity Planning

High-speed cornering during racing causes significant tire load transfer on each wheel due to a lateral acceleration at the roll axis. Since the tire load governs the maximum performance of the tire, a model-based velocity strategy accounting for the real-time wheel load is necessary to maximize the tire performance without losing tire grip. We introduce a dynamics-aware velocity planning algorithm that derives the velocity plans with maximum tire performance based on the learned tire dynamics. We first compute the real-time vertical tire load  $F_{z,i}$  affected by the lateral load transfer  $\Delta W_f$  [17]. The diagram of the load transfer at the roll axis is illustrated in the left of Fig. 2. The load transfer is computed by the roll couple  $C_{roll} = m_s \dot{v}_y h_a$ , where  $m_s$  is the sprung mass,  $\dot{v}_y$  is the lateral acceleration, and  $h_a$  is the roll height. As the learned tire model describes the characteristic for the nominal tire load  $\bar{F}_{z,i}$ , we compute the maximum lateral force of each tire  $F_{y,i}^{max}$  in terms of the tire load ratio with peak value of the tire model as:

$$F_{y,i}^{max} = \mu \frac{F_{z,i}}{\bar{F}_{z,i}} F_{y,i}^{peak}, \quad (10)$$

<sup>1</sup>The coefficients  $C_d, C_r$  were provided by an IAC official.

<sup>2</sup>Dyno data refers to the data collected during engine testing on a dynamometer.

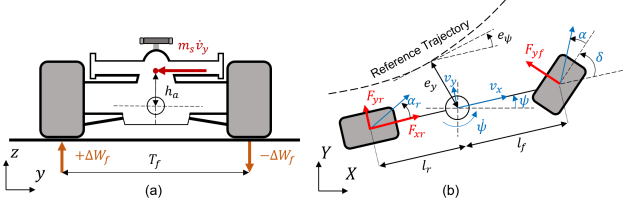


Fig. 2. **Left:** Lateral load transfer generation by the lateral acceleration at the roll axis. **Right:** Overall diagram of the vehicle model.

where  $\mu$  is a tire performance factor to control the confidence and maximum performance of the tire model,  $\frac{F_{z,i}}{F_{z,i}^{\max}}$  is the tire load ratio. The maximum lateral acceleration is determined by the following lateral motion dynamics [13]:

$$a_{y,\max} = \frac{1}{m} (F_{y,r}^{\max} + F_{y,f}^{\max} \cos(\delta) - m v_x \dot{\psi}), \quad (11)$$

where  $\delta$  is the steering angle and  $\dot{\psi}$  is the yaw rate. Then a desired maximum velocity  $v_{x,\text{des}}$  is planned according to the curvature  $\kappa$  of a reference path from a planning module [18]:

$$v_{x,\text{des}} = \sqrt{a_{y,\max} / \kappa}. \quad (12)$$

### B. Throttle and Brake Control

The planned desired velocity is fed into a feedback control module [19] to compute the traction force. However, as shown in Fig. 1, another low-level controller to transform the traction force to the throttle command is required to control the racecar with nonlinear driveline dynamics. We design the throttle and brake control system following [20]. Fig. 3 shows the details of the low-level control system. We exploit the integrated engine torque map to convert the desired engine torque  $T_{e,\text{des}}$  to the desired throttle command  $\tau_{t,\text{des}}$ . As the torque map is built as a lookup table, we search the desired throttle with respect to a given engine speed and desired torque. The inverse brake model is a module to convert the braking force to the brake pedal command, which is activated if  $F_{x,r}$  is negative. The brake model is also attained by our proposed MI-HPO, but details are omitted to conserve space.

### C. Model-based Path Tracking Control

We follow the lateral vehicle dynamics of [14] illustrated in the right of Fig. 2. The lateral model is derived from the objective of tracking a reference trajectory. We implement path tracking control by stabilizing a velocity-dependent chassis model in terms of the error state variables  $\xi$  and control  $u$ .

$$\xi = [e_y, \dot{e}_y, e_\psi, \dot{e}_\psi]^T, \quad u = \delta, \quad (13)$$

where  $e_y, e_\psi$  denote the position and orientation error with respect to a given trajectory. The lateral model contains tire-related model parameters such as the cornering stiffnesses of the front and rear tires  $C_{\alpha,f}, C_{\alpha,r}$ . Since the lateral dynamics is obtained from the bicycle model, the front and rear tire models are optimized according to Eq. 4, but the sum of the left and right tire forces is used as the model output. The cornering stiffnesses then can be approximated as follows [14]:

$$C_{\alpha f} \approx B_f \times C_f \times D_f, \quad C_{\alpha r} \approx B_r \times C_r \times D_r. \quad (14)$$

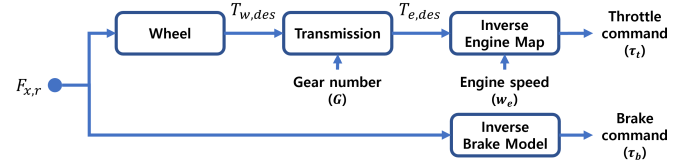


Fig. 3. Throttle and brake control system.

Based on the lateral model, we design the Linear Quadratic Regulator [1] with the following optimization problem:

$$\min_u \int_0^\infty (\xi^T Q \xi + u^T R u) dt, \quad (15)$$

where  $Q, R$  denote gain matrices for LQR. For the real-time control performance, we compute the state feedback optimal LQR gains over piecewise velocity intervals offline [21].

## V. EVALUATION

### A. Analysis for Model Parameter Identification

1) *Tire Dynamics Model:* Fig. 4 illustrates the learned tire models with datasets provided as the tire property files (\*.tir)<sup>3</sup>. The files contain tire force and moment characteristics with high fidelity [22]. To identify the model, we sampled 3000 data for each tire using the property files in various tire state conditions such as tire load, camber angle, slip angle, and slip ratio. As depicted in the left and middle of Fig. 4, the learned models show good fitness to the tire characteristic distribution. We further investigated the tire model with the test drive data collected during track racing. The right of Fig. 4 illustrates the front tire model of the single-track bicycle dynamics learned by the provided tire property data comparing it with the driving data that is not used for learning. The learned model shows the generalization ability for the overall data distribution represented by blue dots. However, since we obtained the model by offline optimization and focused on the representativeness of data, the model needs higher accuracy in some edge cases near the peaks of the lateral force. To handle these cases, we can create an online parameter optimization scheme by parallelizing the HPO process [9] in MI-HPO, which we will implement in future work.

2) *Engine Torque Model:* Fig. 5 illustrates the learned engine torque curves and integrated engine map. The data for the engine map was provided by engine dynamometer testing. For higher reliability, we incorporated our data-driven engine torque models with the dyno data, especially for the throttle pedals 5, 15, and 20%, where the dynamometer showed insufficient accuracy in torque measurements. The result shows that the learned torque curves are able to represent the change in the maximum torque according to the throttle commands. Moreover, the learned models also fit the torque curves that change nonlinearly as a function of engine speed. We integrated the learned models with the provided dyno data and interpolated the torque data to construct an engine lookup table. The blue area on the right of Fig. 5 shows the interpolated region by the learned torque curves. Our vehicle utilized the learned region in racing scenarios such as pit-in/out, obstacle avoidance, and driving within 100 km/h.

<sup>3</sup>The property files for the four tires were provided by an IAC official.



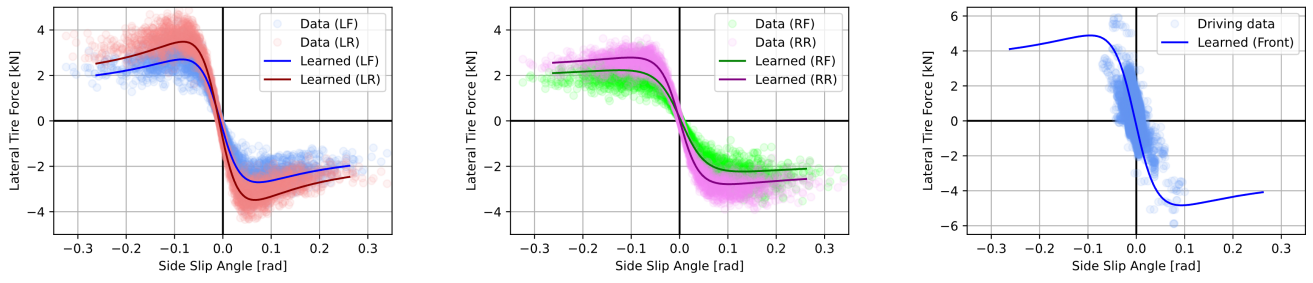


Fig. 4. Learned tire models with the provided tire data (Left: left-front and right-front, Middle: left-rear and right-rear). Right: Learned front tire dynamics of the single-track bicycle model and the distribution of the collected data on the track.

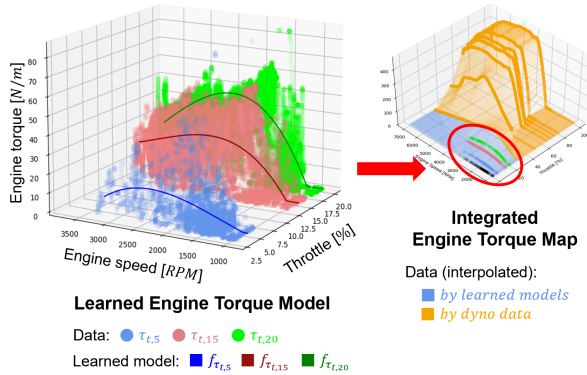


Fig. 5. Learned engine torque curves and the integrated engine map.

**3) Comparison Study:** To evaluate our method's performance, we compared it with four baseline algorithms based on GBO and PSO. We configured two GBO-based methods, namely GBO-small and GBO-large, with learning rates of  $5 \times 10^{-12}$  and  $1 \times 10^{-10}$ , respectively. Additionally, we constructed PSO-100 and PSO-500 with 100 and 500 particles, respectively, utilizing the default acceleration coefficients defined in [5]. For the MI-HPO algorithm, we set  $R = 1 \times 10^4$  and  $\eta = 5$ . To evaluate the algorithms' performance, we conducted parameter identification for the left-front tire model using the data shown in Fig. 4. All algorithms had the same initial configurations. The average model error curves over time obtained from the evaluation are presented in Fig. 6. Our method outperformed other methods by being 13 and 494 times faster in achieving an error of 1.0 kN and 0.5 kN, respectively. Furthermore, the MI-HPO method achieved a terminal error of 0.39 kN, resulting in a 24% and 76% better convergence than PSO-500 and GBO-large methods, respectively. These results demonstrate the effectiveness of our algorithm in exploring a large number of configurations initially ( $n = 1 \times 10^4$ ) and efficiently exploiting well-performing candidates by discarding those with poor performance. GBO-small, with its stable but small learning rate setting, demonstrated negligible improvement in parameter values. Although GBO-large showed better performance than GBO-small, it yielded an unstable optimization process with convergence to a local minimum. PSO-100 had fast initial parameter updates, but it failed to find more optimal parameters beyond an error of 0.75 kN. On the other hand, PSO-500 found a better solution than PSO-100 with more particles, but required higher computational resources due to the increased number of particles. As a result, it had a significantly slower convergence rate compared to our method.

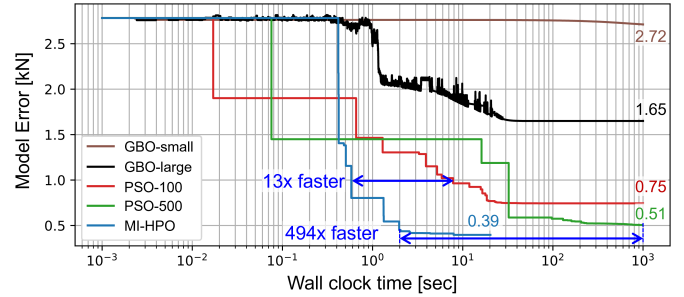


Fig. 6. Average model error curves of different evaluation algorithms with the final model error corresponding to each color-coded number.

## B. Control Performance in Indy Autonomous Challenge

Our model-based planning and control algorithms were deployed in the full-scale racecar platform. Moreover, we extensively validated our learned parameter-based algorithms in the real-world tracks, IMS and LVMS. Our vehicle successfully performed various scenarios, such as obstacle avoidance and high-speed racing over 200 km/h on the tracks.

**1) Obstacle Avoidance at the IMS:** Fig. 7 shows the quantitative results of an obstacle avoidance mission on the front stretch of the IMS. In this mission, obstacles are located before the first-corner section, where the velocity plan is critical for avoiding collision while keeping close to the racing line. For the sake of safety, we set the tire performance factor  $\mu$  as 0.7. Our dynamics-aware velocity planner was able to allow the racecar to maximize the velocity while regulating the lateral acceleration within the learned maximum tire performance during the rapid avoidance maneuvers. The obstacle avoidance was initiated in high-speed driving at 100 km/h, and steering commands were computed up to  $-10^\circ$  to follow the generated collision-free reference trajectory. The sharp steering command could cause significant lateral acceleration higher than  $9.0 \text{ m/s}^2$ , which might cause critical tire grip loss. However, our tire model-based velocity planner was able to plan a safe desired speed that did not excessively increase lateral acceleration, and thereby prevent significant loss of tire grip, by inferring a reasonable maximum lateral acceleration based on the real-time tire loading.

**2) High-Speed Autonomous Driving in LVMS:** Furthermore, we extensively validated the control performance based on the optimized model parameters at LVMS. Fig. 8 illustrates the quantitative results of the lateral and longitudinal control while our vehicle raced more than nine laps (23 km). Our path-tracking algorithm shows robust control performance leveraging the learned tire parameters. The largest position

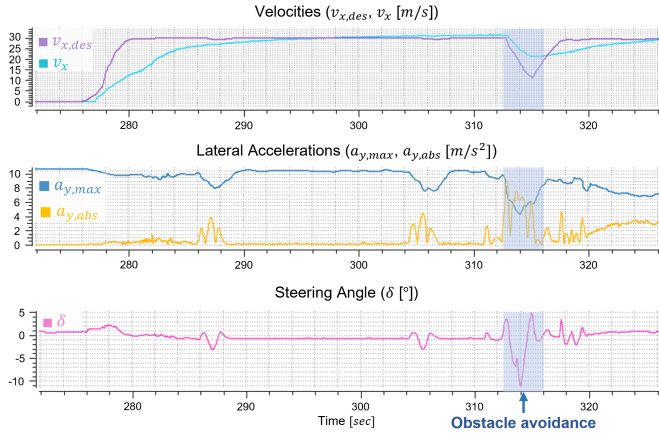


Fig. 7. Results of the velocity control, lateral accelerations, and steering angles during the obstacle avoidance mission at the IMS.

and orientation errors were 0.6 m and  $-2.2^\circ$ , respectively. In addition, the AV-21 succeeded high-speed autonomous driving at above 144 km/h (with a top speed of 217.4 km/h), where the dynamic scenario had yet to be visited and adjusted before this track experiment. These results demonstrate that MI-HPO can optimize and provide appropriate prior dynamics models offline for the design of model-based control before deployment. However, after the vehicle surpassed 144 km/h, our low-level controller computed throttle commands of over 40%, where the engine range was based solely on the provided dyno data, without our data-driven identification. This could have contributed to the velocity error observed in the range of 151–187 km/h (42–52 m/s in Fig. 8). Nevertheless, the control system can be improved with an extended data-driven engine map for throttle control at high-speed. We also point out that our method has the potential to be processed online by parallelizing the HPO process [9], which enables the method to identify the model in real-time during deployment.

## VI. CONCLUSION

We present MI-HPO, a method for identifying model parameters by utilizing a hyperparameter optimization scheme. Our approach showed the ability to optimize the parameters of the dynamics models, such as the tire models and engine torque curves, and exhibited better convergence performance than the traditional optimization schemes. Furthermore, the model-based planning and control system with the learned model parameters demonstrated stable performance in the real-world track environments, IMS and LVMS. In future works, we will implement the online HPO method and integrate it with the offline method of this work to iteratively infer the changing parameters of the vehicle dynamics while on track.

## REFERENCES

- [1] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012. 1, 4
- [2] C. Jung, S. Lee, H. Seong, A. Finazzi, and D. H. Shim, “Game-theoretic model predictive control with data-driven identification of vehicle model for head-to-head autonomous racing,” *arXiv preprint arXiv:2106.04094*, 2021. 1
- [3] A. K. Tangirala, *Principles of system identification: theory and practice*. Crc Press, 2018. 1

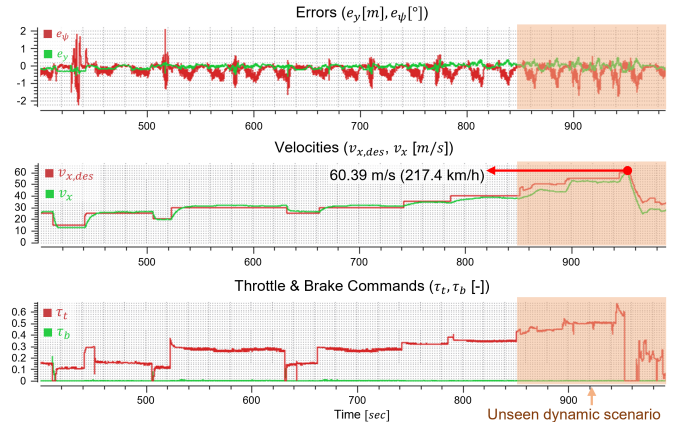


Fig. 8. Results of the position/orientation errors, velocity control, and throttle/brake controls in LVMS.

- [4] E. K. Chong and S. H. Zak, *An introduction to optimization*. John Wiley & Sons, 2013, vol. 75. 1
- [5] P. Erdoğan and S. Ekiz, “Nonlinear regression using particle swarm optimization and genetic algorithm,” *International Journal of Computer Applications*, vol. 153, no. 6, pp. 28–36, 2016. 1, 5
- [6] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelmann, and J. C. Gerdes, “Neural network vehicle models for high-performance automated driving,” *Science robotics*, vol. 4, no. 28, p. eaaw1975, 2019. 1
- [7] L. Hermansdorfer, R. Trauth, J. Betz, and M. Lienkamp, “End-to-end neural network for vehicle dynamics modeling,” in *2020 6th IEEE Congress on Information Science and Technology (CiSt)*. IEEE, 2021, pp. 407–412. 1
- [8] M. Feurer and F. Hutter, “Hyperparameter optimization,” in *Automated machine learning*. Springer, Cham, 2019, pp. 3–33. 1
- [9] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017. 2, 4, 6
- [10] Energy Systems Network, “Indy autonomous challenge,” 2022. [Online]. Available: [www.indyautonomouschallenge.com](http://www.indyautonomouschallenge.com) 2
- [11] S. Mirjalili, “Genetic algorithm,” in *Evolutionary algorithms and neural networks*. Springer, 2019, pp. 43–55. 2
- [12] E. Bakker, L. Nyborg, and H. B. Pacejka, “Tyre modelling for use in vehicle dynamics studies,” *SAE Transactions*, pp. 190–204, 1987. 3
- [13] J. Kabzan, M. I. Valls, V. J. Reijgwart, H. F. Hendriks, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan *et al.*, “Amz driverless: The full autonomous racing system,” *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020. 3, 4
- [14] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011. 3, 4
- [15] L. Li, Z. Zhu, X. Wang, Y. Yang, C. Yang, and J. Song, “Identification of a driver’s starting intention based on an artificial neural network for vehicles equipped with an automated manual transmission,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 230, no. 10, pp. 1417–1429, 2016. 3
- [16] J. S. Killedar, *Dynamometer: theory and application to engine testing*. Xlibris Corporation, 2012. 3
- [17] D. Seward, *Race car design*. Bloomsbury Publishing, 2017. 3
- [18] D. Lee, C. Jung, A. Finazzi, H. Seong, and D. H. Shim, “Resilient navigation and path planning system for high-speed autonomous race car,” *arXiv preprint arXiv:2207.12232*, 2022. 4
- [19] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback control theory*. Courier Corporation, 2013. 4
- [20] K. Hedrick *et al.*, “Brake system modeling, control and integrated brake/throttle switching phase i,” 1997. 4
- [21] J. Spisak, A. Saba, N. Suvarna, B. Mao, C. T. Zhang, C. Chang, S. Scherer, and D. Ramanan, “Robust modeling and controls for racing on the edge,” *arXiv preprint arXiv:2205.10841*, 2022. 4
- [22] A. Schmeitz, “Mf-tyre/mf-swift,” *Delft: TNO*, 2013. 4