Beyond Constant Parameters: Hyper Prediction Models and HyperMPC

Jan Węgrzynowski[1,2], Piotr Kicki[1,2], Grzegorz Czechmanowski[1,2] Maciej Krupka[1], Krzysztof Walas[1,2]

***Abstract*—Model Predictive Control (MPC) is among the most widely adopted and reliable methods for robot control, relying critically on an accurate dynamics model. However, existing dynamics models used in the gradient-based MPC are limited by computational complexity and state representation. To address this limitation, we propose the Hyper Prediction Model (HyperPM) - a novel approach in which we encode the unmodeled dynamics onto a time-dependent dynamics model. This time-dependency is captured through time-varying model parameters, whose evolution over the MPC prediction horizon is learned using a neural network. Such formulation preserves the computational efficiency and robustness of the base model while equipping it with the capacity to anticipate previously unmodeled phenomena. We evaluated the proposed approach on several challenging systems, including real-world F1TENTH autonomous racing, and demonstrated that it significantly reduces long-horizon prediction errors. Moreover, when integrated within the MPC framework (HyperMPC), our method consistently outperforms existing state-of-the-art techniques.**
Full Paper, Videos, and code are available at hyper-mpc.github.io

***Index Terms*—Dynamics Model Learning, Model Predictive Control, MPC**

I. Introduction

One of the fundamental challenges in robotics is determining optimal actions to achieve specific objectives, a task generally referred to as control. This challenge is particularly pronounced for systems that are underactuated or operate at the limits of their physical capabilities. Many of these complexities have been recently resolved with learning-based controllers [1], [2], [3], [4]. However, if the model of the system is available, Model Predictive Control (MPC) has demonstrated outstanding performance across demanding tasks, such as agile drone flight [5], [6], [7], autonomous racing [8], [9], and legged locomotion [10], [11].

MPC methods can be broadly categorized into sampling-based approaches, such as MPPI [12], [13], [14], [15], which offer flexibility at the expense of computational demand, and optimization-based approaches [16], which often leverage physics-derived models [11], [17] or small function approximators [8], [7], [18]. Nevertheless, the effectiveness of all these approaches depends fundamentally on the quality of the underlying dynamics model.

Dynamics models in MPC span a spectrum: reduced-order analytical models derived from first principles [19], [20], purely data-driven neural networks [21], [22], [23], and hybrid approaches combining physics-based models with Gaussian Processes [24], [25], [26], polynomial expansions [7], [27], or neural networks [28], [29], [30]. However, all approximators are limited by partial observability i.e. state representation. Including history can mitigate this issue [28], [23], but recurrent models [31], [32] expand the system state, resulting in cubic growth in MPC complexity. Hyper-networks [33], [34], [35] and parameter-predicting neural networks [9] offer intriguing alternatives, though they assume static parameters over the MPC horizon.

To address these limitations, we propose the Hyper Prediction Model (HyperPM), which encodes the unmodeled components of the system dynamics as time-dependent variations in the parameters of the existing model. This way, we can enhance the accuracy of the dynamics model without affecting its computational complexity. We implement HyperPM as a neural network that predicts the trajectories of model parameters conditioned on past states, controls, and future planned inputs (see Fig. 1). Unlike prior works [35], [9], which assume time-invariant parameters, HyperPM anticipates their evolution, yielding lightweight yet accurate long-horizon dynamics models. Furthermore, we propose HyperMPC, which integrates HyperPM with MPC, exploiting the synergy between superb long-horizon prediction accuracy of HyperPM and planned control trajectories obtained with MPC.

We evaluate HyperPM on two challenging tasks: (i) rope-suspended payload tracking with a drone, and (ii) real-world F1TENTH racing [36]. HyperPM improves prediction accuracy by over 12%, and 62% in these tasks compared to models with constant parameters, while HyperMPC improves control performance by up to 9%, and 19%, respectively.

The contributions of this paper are the following:
1) Hyper Prediction Model, a novel framework that predicts future trajectories of dynamics model parameters based on history and planned actions,
2) HyperMPC, which integrates HyperPM with MPC to improve control performance without increasing the computational complexity of the dynamics model.

II. Proposed Method

A. Hyper Prediction Model

To motivate our approach, consider using MPC to control a drone carrying a rope-suspended payload, where the observable state includes only the drone state variables - position, orientation, and linear and angular velocities. Therefore, a system's model relying solely on (x_t, u_t) will inevitably incur errors, because the influence of the swinging payload cannot be accurately captured without access to its unobserved state. Moreover, MPC requires knowledge about the payload's

[1]Institute of Robotics and Machine Intelligence, Poznan University of Technology, Poland, [2]IDEAS Research Institute, Warsaw, Poland,
corresponding author: jan.wegrzynowski@put.poznan.pl

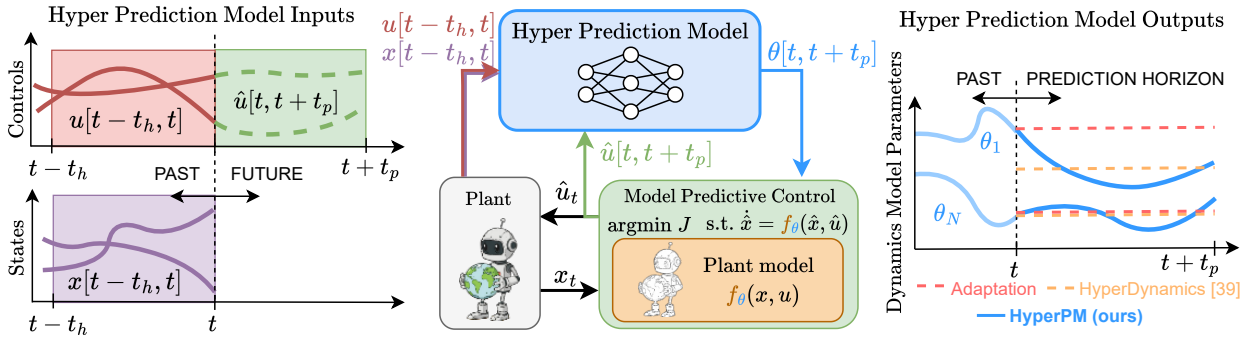

Fig. 1: **HyperPM** leverages the recent state $x[t_c-t_h,t_c]$ and control $u[t_c-t_h,t_c]$ history, together with the planned controls $\hat{u}[t_c,t_c+t_p]$, to predict a trajectory of time-varying model parameters $\theta[t_c,t_c+t_p]$. Injecting these predicted parameters into the nominal dynamics yields more accurate long-horizon state forecasts, allowing **HyperMPC** to proactively anticipate and compensate for previously unmodeled effects.

influence not just at the current time step, but throughout the entire prediction horizon.

To provide the dynamics model with the ability to compensate for unmodeled phenomena along the entire prediction horizon, we propose to encode these effects into the variations of the dynamics model parameters. We challenge the common assumption of time-invariant model parameters and propose to infer the trajectory of unmodeled states over the prediction horizon. The impact of these unmodeled states is then captured as a trajectory of time-varying model parameters $\theta[t_c,\ t_c+t_p]$, yielding the Hyper-Prediction Model (HyperPM):

$$\theta[t_c,\ t_c+t_p] = \text{HyperPM}_\Theta\left(\begin{matrix} x[t_c-t_h,\ t_c] \\ u[t_c-t_h,\ t_c] \end{matrix}, \hat{u}[t_c,\ t_c+t_p]\right), \tag{1}$$

which ingests the recent state history $x[t_c-t_o,t_c]$, the corresponding control history $u[t_c-t_o,t_c]$, and the planned control sequence $\hat{u}[t_c,t_c+t_p]$, to predict the expected trajectory of time-varying model parameters $\theta[t_c,\ t_c+t_p]$. The scheme of HyperPM is presented in Figure 1. The HyperPM model (i) extracts a latent representation of the unobserved unmodeled state from history of observed states and actions, (ii) propagates that representation forward under the planned actuation, and (iii) expresses the resulting influence of unmodeled states as a trajectory of time-varying parameters $\theta[t_c,\ t_c+t_p]$ across the prediction horizon. Additional implementation details may be found in the full paper referenced in the abstract.

Note that our approach goes beyond adapting the dynamics model's parameters, as approaches of this type assume that adapted parameters are constant over the prediction horizon. Interestingly, even an ideal adaptation can be outperformed by HyperPM due to its capabilities to capture expected future effects of unmodeled dynamics, as shown in the experimental analysis (see Section III-B).

In summary, our method addresses scenarios where obtaining a fully Markovian representation of the system state is infeasible or computationally prohibitive. By approximating certain missing aspects of the dynamics within the model's parameter space, HyperPM improves the accuracy of long-horizon predictions by encoding unmodeled dynamics into trajectories of time-varying parameters.

B. *Training procedure*

Training HyperPM requires a tailored procedure, as its goal of predicting a trajectory of model parameters $\theta[t_c,t_c+t_p]$ makes conventional single-step prediction losses unsuitable. Instead, we predict the parameter's trajectory and use it to roll out the time-varying dynamics f_{θ_t}, generating a predicted state trajectory $\hat{x}[t_c+dt,t_c+t_p]$. Then, we compute loss as mean squared error between this predicted state trajectory and the ground-truth data. We optimize the HyperPM weights Θ by back-propagating the gradients of this trajectory-wise loss through time [37]. This end-to-end approach directly optimizes for the long-horizon prediction accuracy, which is crucial for MPC.

C. *HyperMPC*

The downstream application of HyperPM, which we focus on in this paper, is optimization-based Model Predictive Control [16], [38]. In this framework, an optimization algorithm computes trajectories of states and control actions that minimize the objective function, while satisfying the constraints stemming from the robot's dynamics and imposed on its states and control signals. In order to close the feedback loop, only the first control action is applied to the plant, and the whole optimization process is repeated at the next control interval.

The key aspect of the MPC paradigm is the use of a model capable of accurately predicting the evolution of the system over the optimization horizon. Typically, MPC uses a single constant model of the system [16]. However, as shown in [35], it is possible to infer a new model at each control interval, so that it can capture local variations in the dynamics of the system. In our approach, called HyperMPC, we propose to go further and exploit HyperPM to predict the trajectory of model parameters over the MPC horizon and increase predictive accuracy not only locally but also in the expected near future.

We obtain a horizon-long trajectory of dynamics-model parameters using HyperPM that is fed with both the recent history of observations and the planned control sequence generated by the previous MPC iteration. These predicted parameters are then injected into the nominal dynamics model and supplied to the MPC solver, which resolves the optimal-control problem for the current horizon. Only the first action

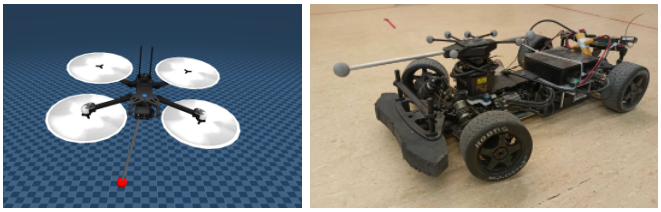

Fig. 2: Experimental platforms used to validate the proposed method: (i) drone with rope attached payload trajectory tracking, (ii) real-world autonomous F1TENTH [36] racing.

of the refreshed solution is applied to the plant; the state and planned-action buffers slide forward, and the entire cycle repeats at the next control interval.

Crucially, the scheme maintains the same computational complexity as the primary dynamics model within the optimization loop, adding only a single forward pass through a lightweight neural network performed before solving the optimal control problem.

III. Experiments and Results

To evaluate the effectiveness of the proposed HyperPM and HyperMPC, we performed experiments on two challenging control tasks using systems presented in Fig. 2. We show that forecasting the trajectories of dynamic model parameters using HyperPM enhances the prediction accuracy and improves the control performance in MPC. In each experiment, control performance is evaluated via the accumulating MPC stage cost across episodes, highlighting differences between the modeling approaches.

A. Baselines

To establish baselines for our method, we selected a group of approaches to model the dynamics of the considered systems: **const**$_s$ – a dynamics model with constant parameters, obtained using single-step prediction error minimization. **const**$_l$ – a dynamics model with constant parameters, optimized using long sequences to minimize the error between simulated system states and ground-truth states across entire trajectories. **HD**$_s$ – referred to as HyperDynamics, inspired by [9], [35]. This is a dynamics model in which parameters are predicted by a neural network and held constant for the entire prediction horizon of the roll-out. It is trained using single-step prediction error minimization according to [9], [35]. **HD**$_l$ – the same architecture as HD$_s$, but trained using long sequences.

In addition, we tested extending the nominal dynamics of the system with a residual neural network [29], [30], $\dot{x} = f(x,u) + \mathrm{NN}(x,u)$, trained using long sequences, which we refer to as **res**. For drone experiments, the residual network predicts the residual forces acting on the drone frame.

To ensure a fair comparison with baselines, the training and architectural hyperparameters (e.g., neural network architecture, batch size, and learning rate) were optimized for each method by grid search, using the accuracy of the validation subset as the evaluation criterion. Full details of the training setup, including hyperparameters, exact MPC formulation, and dataset descriptions, are provided in the full paper.

TABLE I: Long-horizon prediction error for drone with payload. Improvement is calculated w.r.t. const$_l$ model with adequate dynamics model.

Nominal Model	Error (mean ± std) (↓)	Improvement [%] (↑)
const$_l$	0.0655 ± 0.132	—
HD$_l$	0.0252 ± 0.083	61.52
HyperPM (ours)	$\mathbf{0.0176} \pm 0.076$	**73.12**
Residual Model	**Error (mean ± std) (↓)**	**Improvement [%] (↑)**
const$_l$	0.0186 ± 0.058	—
HD$_l$	0.0184 ± 0.057	1.08
HyperPM (ours)	$\mathbf{0.0164} \pm 0.060$	**11.82**

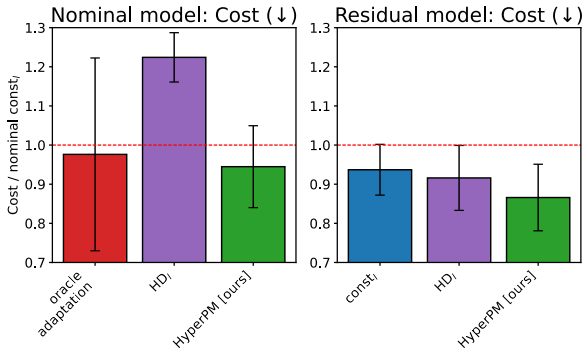

Fig. 3: Comparison of drone modeling approaches for an MPC trajectory tracking task. Values are relative to the nominal const$_l$ model.

B. Simulated drone with rope suspended payload

Drone trajectory tracking is a fundamental component of an aerial vehicle's autonomy stack and presents a particularly challenging modeling problem, especially when the drone is tasked with transporting a rope-suspended payload. In our setup, the rope was attached to the center of mass of the drone frame, allowing us to model the behavior using only forces. It is important to note that the state of the system does not include the position or velocity of the payload. This partial observability introduces complex, time-varying disturbances not evident from the drone's state alone, making accurate long-horizon prediction and control challenging.

In our experiments, we used two dynamics models, nominal [39], and a residual model in which a neural network predicts drone frame residual forces. Dynamics model parameters that can be changed by HD$_l$ or HyperPM are the additional forces acting on a drone's frame. Additionally, we include an oracle-adaptation baseline that retrieves the exact payload-induced forces from the simulator and assumes that these forces remain constant throughout the prediction horizon. This represents an idealized scenario with access to perfect information, albeit with the limitations of time-invariant parameterization. We omitted the models trained using single-step prediction as they proved unstable inside the MPC loop and failed to deliver reliable control performance.

The reported results are based on a setup that employs a 0.5 kg payload suspended from a 1 m rope attached to a quadrotor of mass 1.325 kg. In the long-horizon prediction task (1s horizon, 100Hz), both HD$_l$ and HyperPM substantially im-

TABLE II: Long-horizon prediction error for F1TENTH car

Model	Error (mean ± std) (↓)	Improvement [%] (↑)
const_s	0.0240 ± 0.0581	0.00
const_l	0.0177 ± 0.0290	26.25
HD_s	0.0501 ± 0.1880	-108.75
HD_l	0.0137 ± 0.0239	42.92
res	**0.0091** ± 0.0138	**62.08**
HyperPM (ours)	0.0123 ± 0.0192	48.75

TABLE III: Comparison of MPC solve time, and parameter inference time on mobile AMD Ryzen 5 4600HS CPU.

Model	Solve time [ms] (↓) (mean ± std)	Inference time [ms] (↓) (mean ± std)
const_s	**10.76** ± 2.24	–
const_l	**10.53** ± 1.97	–
HD_s	13.42 ± 3.53	1.56 ± 0.13
HD_l	**10.65** ± 2.05	1.48 ± 0.14
res	23.73 ± 4.76	–
HyperPM (ours)	**10.61** ± 1.76	1.85 ± 0.12

prove upon the nominal model, yet HyperPM's performance is notably better. However, for residual dynamic modeling, only HyperPM delivers a clear gain, exceeding all other methods by more than 10%, underscoring its ability to anticipate complex, time-varying unmodeled effects. To evaluate the practical utility of our approach, we tested how the learned dynamics models perform in a trajectory-tracking task using MPC following the formulation in [39]. For referenced trajectory we used the test split of the generated dataset. As presented in Figure 3, our approach surpasses both the oracle-adaptation baseline and HD_l in both the nominal and residual dynamics model settings. These results further confirm the advantage of modeling time-varying parameter trajectories in capturing complex, unobserved dynamics and translating that capability into superior control performance.

C. F1TENTH racing

Autonomous racing represents a unique and highly demanding testbed for evaluating and advancing control algorithms. Unlike conventional autonomous driving scenarios, racing pushes vehicles to operate at the edge of their dynamic limits, requiring algorithms to manage highly nonlinear dynamics and unrecoverable consequences of actions. The subsequent experiments use models trained on a 36-minute dataset collected using a real F1TENTH vehicle, with expert drivers manually navigating a variety of racetracks to span the entire operational envelope of the vehicle. Our model-based controller objective is to maximize track progress while keeping the vehicle within track boundaries.

In our experimental analysis, we first evaluate the prediction performance of the models on long-horizon sequences drawn from the manual driving test set. In Table II, we present the prediction errors for all evaluated modeling approaches and refer them to the most classical one – const_s. The best-performing model is the residual model, which utilizes the ability to model residual errors of the analytical model with a neural network. However, this approach takes advantage of

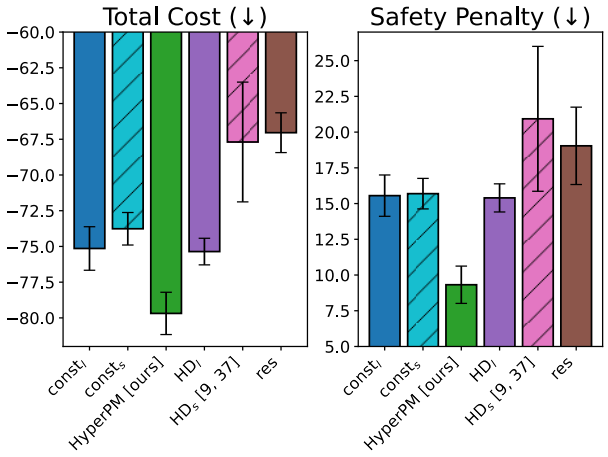

Fig. 4: Comparison of modeling approaches in MPC applied to the real-world F1TENTH racing task.

significantly more parameters than the remaining methods. Among models of the same size, the proposed HyperPM achieves the lowest error, showing a 48.75% improvement over the model with constant parameters.

Finally, we evaluated these modeling approaches in real-world F1TENTH racing. We compared them in 30-second runs using two criteria: (i) total cost $\ell(x, u)$ and (ii) safety penalty $\ell_s(x, u)$, which mainly include track bounds violation soft constraint, both accumulated over the whole run. In Figure 4, we report these metrics averaged over five runs for each method. HyperPM outperforms all methods, including the residual network, which previously had the lowest prediction errors. We attribute this performance gap to the lack of generalization ability of a residual neural network to a different distribution of inputs w.r.t. the manual driving dataset.

Regarding computational cost, Table III shows that feeding a parameter trajectory to the MPC leaves the solver's runtime unchanged.

IV. Conclusion

In this paper, we introduced the Hyper Prediction Model (HyperPM) and its integration into the HyperMPC framework to enhance the predictive accuracy and control performance of systems with complex dynamics. Our method goes beyond the reactive paradigm of existing approaches [9], [35], adaptation schemes, as well as online learning in which model parameters are frozen across the prediction horizon. The proposed framework, by forecasting time-varying parameters of a dynamics model, addresses the limitations of dynamics models, where state representation is often insufficient to describe all important phenomena. The experimental evaluations demonstrated that HyperPM significantly improves the accuracy of the long-term prediction. Moreover, the HyperMPC delivered superior performance in downstream control tasks, outperforming state-of-the-art methods while maintaining the same optimization loop complexity. Our results highlight the importance of incorporating anticipated actions into the prediction of model parameter trajectories, allowing MPC to proactively anticipate and compensate for previously unmodeled effects.

REFERENCES

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] Z.-P. Jiang, T. Bian, and W. Gao, "Learning-based control: A tutorial and some recent results," *Found. Trends® Syst. Control*, vol. 8, no. 3, pp. 176–284, 2020.

[3] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. Volume 5, 2022, pp. 411–444, 2022.

[4] G. Czechmanowski, J. Węgrzynowski, P. Kicki, and K. Walas, "On learning racing policies with reinforcement learning," 2025.

[5] Y. Song and D. Scaramuzza, "Policy Search for Model Predictive Control With Application to Agile Drone Flight," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2114–2130, Aug. 2022.

[6] Y. Song, A. Romero, M. Mueller, V. Koltun, and D. Scaramuzza, "Reaching the Limit in Autonomous Racing: Optimal Control versus Reinforcement Learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, Sep. 2023, arXiv:2310.10943 [cs].

[7] M. Krinner, A. Romero, L. Bauersfeld, M. Zeilinger, A. Carron, and D. Scaramuzza, "MPCC++: Model Predictive Contouring Control for Time-Optimal Flight with Safety Constraints," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[8] 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, Oct. 2019.

[9] J. Chrosniak, J. Ning, and M. Behl, "Deep dynamics: Vehicle dynamics modeling with a physics-constrained neural network for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5292–5297, 2024.

[10] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018, conference Name: IEEE Robotics and Automation Letters.

[11] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4730–4737.

[12] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440.

[13] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.

[14] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius, "Sample-efficient cross-entropy method for real-time planning," in *Conference on Robot Learning 2020*, 2020.

[15] P. Kicki, "Low-pass sampling in model predictive path integral control," 2025. [Online]. Available: https://arxiv.org/abs/2503.11717

[16] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, Nov 2021.

[17] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 690–697, 2022.

[18] T. Salzmann, J. Arrizabalaga, J. Andersson, M. Pavone, and M. Ryll, "Learning for CasADi: Data-driven models in numerical optimization," in *Learning for Dynamics and Control Conference (L4DC)*, 2024.

[19] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact hessian based nonlinear mpc algorithm," in *2016 European Control Conference (ECC)*, 2016, pp. 141–147.

[20] 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.

[21] N. Hansen, H. Su, and X. Wang, "TD-MPC2: Scalable, robust world models for continuous control," in *International Conference on Learning Representations (ICLR)*, 2024.

[22] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "Solar: Deep structured representations for model-based reinforcement learning," in *International conference on machine learning*. PMLR, 2019, pp. 7444–7453.

[23] W. Xiao, H. Xue, T. Tao, D. Kalaria, J. M. Dolan, and G. Shi, "AnyCar to anywhere: Learning universal dynamics model for agile and adaptive mobility," *arXiv preprint arXiv:2409.15783*, 2024.

[24] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars," in *2018 European Control Conference (ECC)*. Limassol: IEEE, Jun. 2018, pp. 1341–1348.

[25] 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, Oct. 2019.

[26] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-Driven MPC for Quadrotors," Mar. 2021, arXiv:2102.05773 [cs].

[27] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 2023.

[28] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Science Robotics*, vol. 4, no. 28, p. eaaw1975, Mar. 2019.

[29] K. Y. Chee, T. Z. Jiahao, and M. A. Hsieh, "KNODE-MPC: A Knowledge-based Data-driven Predictive Control Framework for Aerial Robots," Jan. 2022, arXiv:2109.04821 [cs, eess].

[30] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time Neural-MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, Apr. 2023, arXiv:2203.07747 [cs, eess].

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[33] C. Zhang, M. Ren, and R. Urtasun, "Graph hypernetworks for neural architecture search," in *7th International Conference on Learning Representations (ICLR)*, 2019.

[34] S. Hegde, Z. Huang, and G. S. Sukhatme, "Hyperppo: A scalable method for finding small policies for robotic control," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 10 821–10 828.

[35] Z. Xian, S. Lal, H.-Y. Tung, E. A. Platanios, and K. Fragkiadaki, "HyperDynamics: Meta-learning object and agent dynamics with hypernetworks," in *9th International Conference on Learning Representations (ICLR)*, 2021.

[36] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, and M. Bertogna, "F1/10: an open-source autonomous cyber-physical platform," *CoRR*, vol. abs/1901.08567, 2019.

[37] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990, conference Name: Proceedings of the IEEE.

[38] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[39] B. B. Carlos, T. Sartor, A. Zanelli, G. Frison, W. Burgard, M. Diehl, and G. Oriolo, "An efficient real-time nmpc for quadrotor position control under communication time-delay," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2020, pp. 982–989.