Task-Parameter Nexus for Learning Task-Specific Parameters in Model-Based Control

Sheng Cheng, Yuliang Gu, Ran Tao, Shenlong Wang, Xiaofeng Wang, Naira Hovakimyan

***Abstract*— This extended abstract presents the Task-Parameter Nexus (TPN), a learning-based approach to the online determination of the (near-)optimal control parameters of model-based controllers (MBCs) for trajectory tracking tasks. In TPN, a deep neural network is introduced to generate the control parameters for any given trajectory at runtime. To train this network, we introduce a systematic approach to build the training dataset so that this dataset is rich enough to cover a wide range of trajectories to be tracked. For each trajectory in the bank, we autotune the optimal control parameters offline and use them as labels. With this dataset, the TPN is trained and evaluated on the quadrotor platform. It is shown in simulation experiments that the TPN can predict near-optimal control parameters for a spectrum of tracking tasks, demonstrating its robust generalization capabilities.**

I. Introduction

The recent advancement in large foundational models [1] has improved robots' understanding of their environment [2] and making interactions with humans more natural and effective [3]. This progress is largely due to the availability of large-scale datasets and the ability to learn patterns from vast amounts of visual and textual information, enabling these models to generalize well across different contexts and scenarios [4]. However, limited generalization capability in the domain of planning and control has been reported [4] due to relatively scarce data available for the large models to capture. In addition, even if a large foundation model generalizes well at the level of motion planning, the low-level control (which often follows the model-based design and requires task-specific tuning [5]) may not generalize equally well in terms of the control performance, especially when the tasks are not predefined. These observations lead to a challenging question: Given a task which could be new and not predefined, how can we online determine the associated optimal or near-optimal control parameters in a predefined model-based control structure?

This extended abstract introduces such a tool that enables the system to learn the control parameters when operating different tasks. For the purposes of illustration, we consider a quadrotor platform where a *task* is defined as tracking a trajectory. Different tasks imply different trajectories with various motion characteristics that the quadrotor needs to track, where one single set of control parameters cannot cater to different tasks uniformly well. For instance, consider a PD-type controller for a quadrotor's translational control. If the task requires the quadrotor to hover, then there exists an optimal set of control parameters where the D-gain is dominant: it will allow for sufficient damping so that the quadrotor can hover steadily when it is perturbed. On the contrary, if the task needs the quadrotor to track an aggressive trajectory (high speed or sharp turns), the optimal set of parameters will have a reduced D-gain and raised P-gain (as compared with those for the hover task) for sufficient agility and responsiveness to the fast-changing reference. Since each trajectory to be tracked is potentially associated with a set of optimal control parameters, there exists a nontrivial mapping from tasks to control parameters. We develop a deep neural network, called Task-Parameter Nexus (TPN), together with the training dataset, to approximate this mapping, as shown in Fig. 1. Arbitrarily given a reference trajectory, the TPN predicts the (near-)optimal control parameters of a model-based controller (MBC) for tracking purpose.

This work is supported by NASA cooperative agreement (80NSSC22M0070), NASA ULI (80NSSC22M0070), AFOSR (FA9550-21-1-0411), NSF-AoF RI (2133656), and NSF SLES (2331878, 2331879).

S. Cheng, Y. Gu, R. Tao, S. Wang, and N. Hovakimyan are with the University of Illinois Urbana-Champaign, USA. (email: {chengs, yuliang3, rant3, shenlong, nhovakim}@illinois.edu). X. Wang is with the University of South Carolina, USA. (email: wangxi@cec.sc.edu).

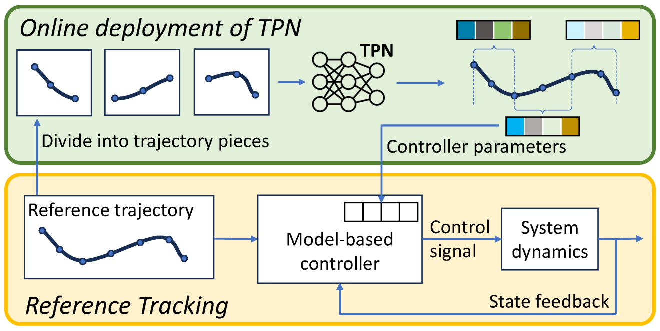

Fig. 1: An illustration of the TPN when applied to a model-based controller for tracking a reference trajectory online.

II. Methodology and Preliminary Results

To train the TPN, we first create a trajectory bank that includes trajectory pieces with various motion characteristics, categorized by speed and curvature. Note that different speeds and curvatures represent different levels of difficulty in tracking the translational and rotational motions, respectively. By changing the speed and curvature (Fig. 2A), the trajectory bank can be enriched to cover a wide range of trajectory pieces the quadrotor may encounter (Fig. 2B). Given the trajectory pieces in the bank, we obtain the expert parameters using batch DiffTune [6]–[8] (Fig. 2C). With the dataset of tasks and expert parameters, we train the TPN in a supervised manner (Fig. 2D). Note that the last layer of the TPN is regulated to output the parameters in the set of stabilizing parameters, which can be determined theoretically

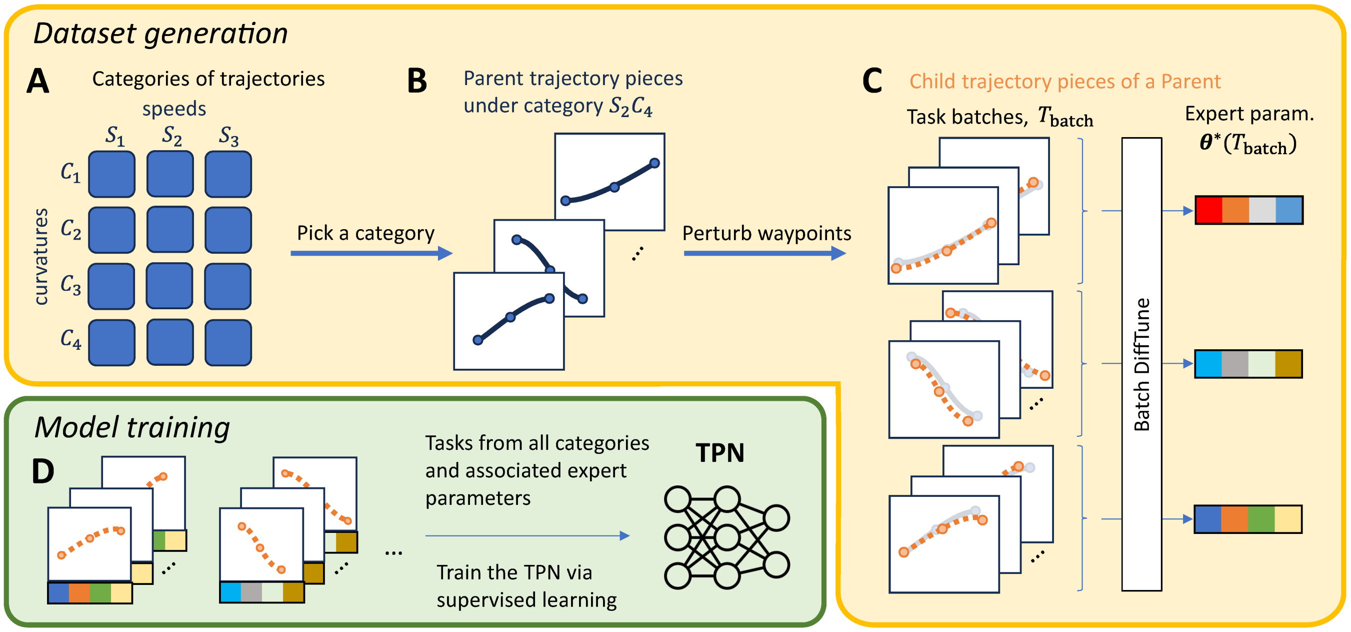
Dataset generation
A
Categories of trajectories
speeds
S_1 S_2 S_3
curvatures
C_1
C_2
C_3
C_4
Pick a category
B
Parent trajectory pieces under category S_2C_4
Perturb waypoints
C
Child trajectory pieces of a Parent
Task batches, $\mathcal{T}_{\text{batch}}$
Expert param. $\boldsymbol{\theta}^*(\mathcal{T}_{\text{batch}})$
Batch DiffTune
Model training
D
Tasks from all categories and associated expert parameters
Train the TPN via supervised learning
TPN

Fig. 2: Illustration of the dataset generation and model training. **A**: Categorize the trajectory bank by speed and curvature. **B**: Generate parent trajectory pieces by sampling waypoints according to the designated speed and curvature and generate smooth trajectories using the minimum snap algorithm. **C**: Randomize trajectory pieces and use batch Difftune over batches of child pieces to obtain the expert (optimal) parameters. **D**: Train the TPN using labeled data generated in **A**–**C**.

or empirically. For example, one can use RAYEN [9] as the last layer if the set is convex. Leveraging the generalization capability of deep neural networks, the TPN can possibly infer appropriate parameters for unseen testing tasks.

When the TPN is deployed online, we first divide the to-be-tracked target trajectory into pieces that match the length of the tasks in the training dataset. So the target trajectory is a collection of sequentially connected trajectory pieces. At runtime, the TPN predicts the control parameters based on each trajectory piece, and the predicted parameter will be loaded to the controller over the time interval corresponding to this piece. The online procedure is illustrated in Fig. 1.

TABLE I: Tracking RMSE achieved by expert parameters, TPN parameters, and untrained parameters over unseen tasks. Unit: [m]

Category	Expert	TPN	Untrained
$\mathcal{S}_1\mathcal{C}_5$	**0.180±0.036**	0.181±0.036	0.249±0.064
$\mathcal{S}_1\mathcal{C}_6$	**0.182±0.038**	0.183±0.038	0.268±0.069
$\mathcal{S}_2\mathcal{C}_5$	0.207±0.059	**0.197±0.057**	0.531±0.089
$\mathcal{S}_4\mathcal{C}_1$	**0.181±0.037**	0.183±0.038	0.268±0.069
$\mathcal{S}_4\mathcal{C}_2$	**0.219±0.070**	0.245±0.074	0.926±0.099
$\mathcal{S}_4\mathcal{C}_3$	**0.218±0.072**	0.245±0.079	1.061±0.097
$\mathcal{S}_5\mathcal{C}_1$	**0.210±0.060**	0.252±0.061	0.532±0.090
$\mathcal{S}_6\mathcal{C}_1$	**0.206±0.060**	0.257±0.065	0.609±0.097

We have validated the TPN in simulation experiments and compared TPN-produced parameters with expert parameters over the tasks (trajectories) under 8 categories that are not covered in the trajectory bank. The results are shown in Table I, in which the TPN's performance, in terms of tracking RMSE, is suboptimal to that of expert parameters under the same categories. Note that the untrained parameters (a single set of stabilizing parameters that were used to initialize the batch auto-tuning) show much worse tracking performance than both the expert and TPN parameters. The overall comparison demonstrates that the TPN enjoys robust generalization capabilities to new tasks.

III. Ongoing and Future Work

We will demonstrate the capability of the TPN on a real quadrotor and develop theoretical support on the generality of the TPN. Extensions of the current setting to different systems will also be explored, such as legged robots.

References

[1] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[2] D. Shah, M. R. Equi, B. Osiński, F. Xia, B. Ichter, and S. Levine, "Navigation with large language models: Semantic guesswork as a heuristic for planning," in *Conference on Robot Learning*. PMLR, 2023, pp. 2683–2699.

[3] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," *IEEE Robotics and Automation Letters*, 2023.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[5] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[6] S. Cheng, M. Kim, L. Song, C. Yang, Y. Jin, S. Wang, and N. Hovakimyan, "Difftune: Auto-tuning through auto-differentiation," *arXiv preprint arXiv:2209.10021*, 2023.

[7] S. Cheng, L. Song, M. Kim, S. Wang, and N. Hovakimyan, "Difftune+: Hyperparameter-free auto-tuning using auto-differentiation," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 170–183.

[8] R. Tao, S. Cheng, X. Wang, S. Wang, and N. Hovakimyan, "DiffTune-MPC: Closed-loop learning for model predictive control," *accepted for publication by IEEE Robotics and Automation Letters*, 2023.

[9] J. Tordesillas, J. P. How, and M. Hutter, "RAYEN: Imposition of hard convex constraints on neural networks," *arXiv preprint arXiv:2307.08336*, 2023.