Proto-MPC: An Encoder-Prototype-Decoder Approach for Quadrotor Control in Challenging Winds

Yuliang Gu   YULIANG3@ILLINOIS.EDU

Sheng Cheng   CHENGS@ILLINOIS.EDU

Naira Hovakimyan   NHOVAKIM@ILLINOIS.EDU

Mechanical Science and Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801

Editors: A. Abate, M. Cannon, K. Margellos, A. Papachristodoulou

Abstract

Quadrotors are increasingly used in the evolving field of aerial robotics for their agility and mechanical simplicity. However, inherent uncertainties, such as aerodynamic effects coupled with quadrotors' operation in dynamically changing environments, pose significant challenges for traditional, nominal model-based control designs. To address these challenges, we propose a multi-task meta-learning method called Encoder-Prototype-Decoder (EPD), which has the advantage of effectively balancing shared and distinctive representations across diverse training tasks. Subsequently, we integrate the EPD model into a model predictive control problem (*Proto*-MPC) to enhance the quadrotor's ability to adapt and operate across a spectrum of dynamically changing tasks with an efficient online implementation. We validate the proposed method in simulations, which demonstrates *Proto*-MPC's robust performance in trajectory tracking of a quadrotor being subject to static and spatially varying side winds.

Keywords: Multi-task Learning, Meta Learning, Model Predictive Control, Aerial Robotics

1. Introduction

In the evolving field of aerial robotics, quadrotors are widely used due to their agility and versatility in various applications. To fully leverage the agility of quadrotors, controller designs are heavily based on quadrotor models. Generally, these models are derived following the Newton-Euler equations, which can hardly accommodate dynamic uncertainties in real-world applications (e.g., wind, aerodynamic effects, slung or slosh payloads). To address this limitation, recent research has focused on using advanced machine learning methods, such as Gaussian Process (Torrente et al., 2021) and NeuralODE (Chee et al., 2022), to learn an accurate dynamical model from real-world data and integrate it with model-based control design, which can significantly enhance the system performance.

Quadrotors operating in real-world scenarios frequently encounter a range of *structurally similar yet appearingly different* tasks, each with unique dynamical uncertainties. For instance, a quadrotor might face varying side wind conditions or be tasked with transporting slung payloads of unknown mass. These varied tasks pose a unique challenge for the above-mentioned control methods. While relying on a single data-driven model often falls short of achieving optimal performance across diverse scenarios, training multiple models for case-specific tasks is inefficient due to 1) challenges in data collection for each specific case and 2) potentially time-consuming online switches of different trained models that use a relatively large amount of parameters for each individual task. To tackle these challenges, a growing line of research investigates the use of online learning and meta-learning techniques. These methods operate in an offline-online framework (O'Connell et al.,
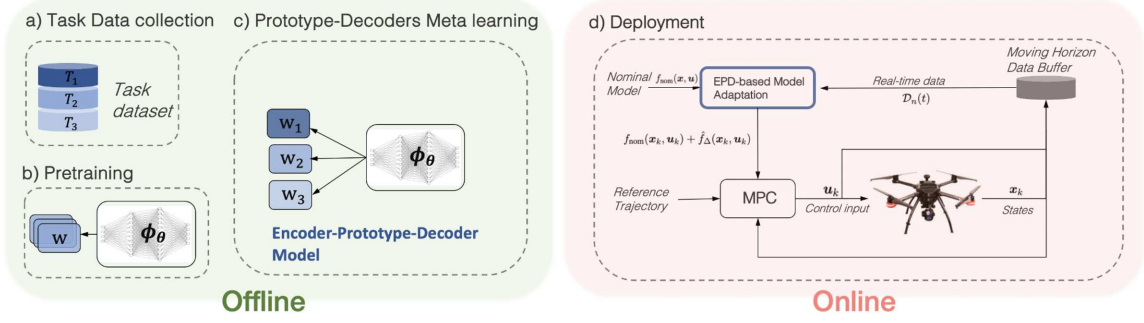
Figure 1: Framework Overview. a) Collecting data on multiple tasks; b) Pretraining to ensure that encoder-decoder pairs can capture the overall patterns of the data; c) Jointly training task-specific prototype decoders to capture distinctive task features and regularizing the encoder to avoid overfitting; d) Online implementation of Proto-MPC with prototype-decoder-based adaptation.

2022; Jiahao et al., 2023; Richards et al., 2021; Wang et al., 2024), allowing for adaptation of the learned models or real-time retraining of new models to align with the changing characteristics of operational tasks. (A comprehensive literature review is available in the Appendix of (Gu et al., 2024).)

Integrating online learning methods into model-based control design poses several key challenges: 1) **adaptivity**: the system must rapidly respond to real-time changing conditions; 2) **model fidelity**: as data-driven models evolve through online learning, they risk losing essential knowledge learned from the initial training data, which can lead to unpredictable behaviors and reduced performance in situations that they were originally designed to handle; 3) **exploration vs. exploitation**: reaching the right balance between exploring new data and exploiting existing knowledge is critical to ensure reliable real-time performance.

To address these challenges, we introduce *Proto*-MPC, a novel multi-task meta-learning-based model predictive control (MPC) framework. Central to our method is an **Encoder-Prototype-Decoder** (EPD) model, which is designed to learn the residual dynamics of the quadrotor from diverse tasks. The EPD model comprises two key components: a universal deep neural network (DNN) encoder and a set of task-specific linear prototype decoders. On the one hand, the encoder learns the common and essential patterns across various task datasets, providing a generalized understanding of the tasks by producing their representations on a low-dimensional manifold (i.e., features). On the other hand, the linear prototype decoder captures the distinctive characteristics of a specific task in a computationally efficient way (due to its linear form). In the online inference stage, the encoder processes incoming data into features, while prototype decoders are used as a "basis" to interpolate encoded features as residuals in the dynamics. This architecture allows fast computation of a new decoder aligned with the current task's characteristics online. Moreover, this adaptive approach ensures the MPC has an accurate, up-to-date residual dynamical model. We evaluate the proposed framework on a quadrotor under various speeds of side wind. The results showcase the generalization and fast adaptation of the proposed *Proto*-MPC framework.

The contributions of this paper are summarized as follows: we propose *Proto*-MPC, a novel model predictive control framework for quadrotor control subject to uncertainties and disturbances. We propose an EPD model as a data-driven augmentation to the physics-based dynamics to capture the uncertainties. The EPD model can achieve the balance between generalizing across a wide array

of tasks, both trained and unseen tasks, and rapidly adapting to dynamically evolving task conditions with tunable parameters.

2. Background: Nonlinear MPC for Quadrotor Control

We consider the 6 DoF rigid body dynamics of the quadrotor (with mass m and inertia J)

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \ \dot{\boldsymbol{v}} = m^{-1}f\boldsymbol{z}_B + \boldsymbol{g}, \ \dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes [0 \ \boldsymbol{\omega}^\top]^\top, \ \dot{\boldsymbol{\omega}} = J^{-1}(\boldsymbol{M} - \boldsymbol{\omega} \times J\boldsymbol{\omega}), \tag{1}$$

where $\boldsymbol{p} \in \mathbb{R}^3$, $\boldsymbol{v} \in \mathbb{R}^3$ stand for the position and velocity of the quadrotor in the inertial frame, $\boldsymbol{q} = [q_0, q^\top]^\top \in \mathbb{S}^3$ (where $q_0 \in \mathbb{R}$ and $q \in \mathbb{R}^3$) is the unit quaternion for rotation from the inertial to body frame, and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity in body frame. The gravitational acceleration is denoted by $\boldsymbol{g}$. The vector $\boldsymbol{z}_B$ is the unit vector aligning with the z-axis of the body frame. The state $\boldsymbol{x} = [\boldsymbol{p}^\top \ \boldsymbol{v}^\top \ \boldsymbol{q}^\top \ \boldsymbol{\omega}^\top]^\top$ follows a discretized version of the dynamics in (1) as $\boldsymbol{x}_{k+1} = f_{\text{nom}}(\boldsymbol{x_k}, \boldsymbol{u_k})$ with control being $\boldsymbol{u} = [f \ \boldsymbol{M}^\top]^\top \in \mathbb{R}^4$ (total thrust f and moment $\boldsymbol{M} \in \mathbb{R}^3$).

The model in (1) describes the nominal case with no dynamical uncertainty. In general, uncertainties (e.g., wind or aerodynamic effects) exist in a real system. We consider lumped uncertainties (see (Wu et al., 2022)), denoted by f_Δ, in the dynamics to account for the impact to the system, resulting in the real dynamics $f_{\text{real}} = f_{\text{nom}} + f_\Delta$. In this paper, we will learn the lumped uncertainties as $\hat{f}_\Delta$. The objective is to ensure that the learned dynamics $f_{\text{nom}} + \hat{f}_\Delta$ closely approximate the actual dynamics f_{real}, which allows us to use it as a trustworthy model in an MPC formulation. We consider the following nonlinear MPC

$$\begin{aligned} \boldsymbol{u}_{0:N-1}^\star = \operatorname*{argmin}_{\boldsymbol{u}_{0:N-1}} \quad & \sum_{k=0}^{N-1} \|\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k\|_Q^2 + \|\boldsymbol{u}_k - \bar{\boldsymbol{u}}_k\|_R^2 + \|\boldsymbol{x}_N - \bar{\boldsymbol{x}}_N\|_{Q_N}^2 \\ \text{subject to} \quad & \boldsymbol{x}_{k+1} = f_{\text{nom}}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \hat{f}_\Delta(\boldsymbol{x}_k, \boldsymbol{u}_k), \ \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \\ & \boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}, \end{aligned} \tag{2}$$

where $\bar{\boldsymbol{x}}_k$ and $\bar{\boldsymbol{u}}_k$ denote the reference state and control, Q and R are the penalty matrices for deviating from the references, and $\boldsymbol{u}_{\min}$ and $\boldsymbol{u}_{\max}$ represent the limits on the control actions.

3. Method

3.1. Dataset

Consider a set of N tasks, $\mathcal{T} = \{T_k\}_{k=1:N}$. We are given their corresponding datasets, $\mathcal{D} = \{D^{T_k}\}_{k=1:N}$, where $D^{T_k} = \{(x, y)\}^{T_k}$ consists of task-specific *identically independently distributed* input-output pairs. The joint distribution of the input-output pairs D^{T_k} is $P^{T_k}(x, y)$. The task-specific batch data (of size n) is $D_n^{T_k}$, which is *uniformly* sampled from D^{T_k}, denoted as $D_n^{T_k} \sim D^{T_k}$, and its empirical distribution is $P_n^{T_k}(x, y)$.

3.2. Prototype-Decoder-Based Meta-Learning

In our approach, we decompose the learned residual dynamics into the following form:

$$y = \hat{f}_\Delta(\boldsymbol{x}, \boldsymbol{u}) = w\phi_\theta(x), \tag{3}$$

where $x = concat[\boldsymbol{x}, \boldsymbol{u}]$ represents the concatenated state and control input vectors, ϕ_θ is an encoder, and w is a linear decoder. Here, the $\phi_\theta \colon \mathbb{R}^{17} \to \mathbb{R}^p$ is a DNN parameterized by θ that encodes input data into a feature space in $\mathbb{R}^p$. The decoder w is a matrix with appropriate dimension and $w \in \mathcal{W} = \{w : \|w\|_2 = \sigma_{\max}(w) < w_0\}$. The decoder maps the encoded features to the output as residuals in the dynamics.

We use the encoder-decoder as shown in (3) to capture the residual dynamics when a quadrotor conducts *structurally similar yet appearingly different* tasks, such as flying in side-wind of different speeds. However, for an encoder-decoder pair with fixed parameters to adapt to different tasks, significant modifications or separate models may be required. To tackle this multi-task scenario, we introduce the EPD model that comprises a task-agnostic encoder ϕ_θ and a set of task-specific prototype encoders $\mathbf{W} = \{\mathbf{w}_k\}_{k=1:N}$. (Note that we use the bold font $\mathbf{w}$ to denote the prototype decoder, which should be distinguished from an arbitrary decoder denoted by w.) On the one hand, the encoder ϕ_θ is trained to be task-agnostic in the sense that it captures the essential characteristics of all task datasets and allows for fast adjustments of the decoder. On the other hand, each prototype decoder $\mathbf{w}_k$ takes the encoded features and outputs precise task-relevant residuals, which essentially fine-tunes the EPD model to operate on the given task T_k. As key components in our method, prototype decoders are used as a "basis" to span a subspace in the task space, which enables 1) offline inter-task regularization and 2) online inter-task interpolation.

3.3. Prototype Decoder

In this subsection, we formally define and derive a prototype decoder. In brief, given an encoder ϕ_θ, a prototype decoder is the most representative of the given task data in some set of decoders. The representativeness of an encoder-decoder pair (ϕ_θ, w) for a task T_k is measured by its empirical risk on task T_k's batch data:

$$\mathcal{R}_n^{T_k}(w, \phi_\theta) = \frac{1}{n} \sum_{(x_i, y_i) \in D_n^{T_k}} \|y_i - \hat{y}_i\|^2, \tag{4}$$

where $\hat{y}_i = w\phi_\theta(x_i)$, and $D_n^{T_k}$ is sampled from D^{T_k}. To ensure that the pair (ϕ_θ, w) captures the overall data patterns effectively, the empirical risk must be bounded in a predefined threshold. We define this property as the *achievability* condition as follows:

Definition 1 *(Achievability) For a task $T_k \in \mathcal{T}$, an encoder-decoder pair (ϕ_θ, w) is achievable with some $R_0 \in \mathbb{R}^+$ if:*

$$\lim_{n \to \infty} \mathcal{R}_n^{T_k}(w, \phi_\theta) = \lim_{n \to \infty} \mathbb{E}_{D_n^{T_k} \sim D^{T_k}} \left[\mathcal{R}(w, \phi_\theta) \right] = \lim_{n \to \infty} \int \mathcal{R}(w, \phi_\theta) dP_n^{T_k}(x, y) \leq R_0. \tag{5}$$

The achievability condition essentially imposes an upper bound on the expected risk to ensure that an encoder-decoder pair has a bounded error over the entire task dataset. One can pretrain the encoder by the alternating minimization method (minimization of the empirical risk by alternating between ϕ_θ and w. More discussions are given in Remark 3) to satisfy the achievability condition. The pretraining step is critical in the sense that the model can learn from data in a "lossy" way while staying anchored to the core features.

Given an encoder ϕ_θ, we define the set of decoders satisfying (5) as a **task-achievable decoder set** $\mathcal{A}_{\phi_\theta}(T_k) = \left\{ w \in \mathcal{W} : \lim_{n \to \infty} \mathcal{R}_n^{T_k}(w, \phi_\theta) < R_0 \right\}$. The task-achievable decoder set $\mathcal{A}_{\phi_\theta}(T_k)$ consequently specifies a task-specific achievable region in $\mathcal{W}$. We are now ready to introduce a

novel component called the ***prototype decoder***, which is a representative in $\mathcal{A}_{\phi_\theta}(T_k)$. The prototype decoder is a critical part of our model, aimed at effectively capturing the individual characteristics of each task:

Definition 2 *(**Prototype Decoder**) For a given task $T^k \in \mathcal{T}$, the prototype decoder, denoted by $\mathbf{w}_k$, achieves the minimal empirical risk over the achievable set: $\mathbf{w}_k = \operatorname{argmin}_{w \in \mathcal{A}_{\phi_\theta}(T_k)} \mathcal{R}_n^{T_k}(w, \phi_\theta)$.*

The prototype decoder captures the central characteristics of its corresponding task to achieve minimal risk among all the achievable decoders. This choice aligns with the principle of risk minimization, focusing on achieving the most efficient and effective learning outcome for each task. In practice, the prototype decoder can be computed empirically via

$$\mathbf{w}_{k,emp} = \operatorname{argmin}_{w \in \bar{\mathcal{W}}} \mathcal{R}_n^{T_k}(w, \phi_\theta), \tag{6}$$

where $\bar{\mathcal{W}}$ is finite set of achievable decoders. This empirical computation results in a geometric interpretation of the role of the prototype decoder: it is the geometric center of the achievable decoders under the "distance" defined by the risk, which is a concept that closely relates to Prototypical Networks ([Snell et al., 2017](#)) for few-shots classification. Similarly the prototype decoder acts as a representative of the associated task in our EPD model framework.

Remark 3 *In Rate-Distortion Theory, the definition of empirical risk in ([4](#)) is in fact a distortion measure between sequences ([Cover, 1999](#)). In our formulation, an achievable decoder set $\mathcal{A}_{\phi_\theta}(T_k)$ with an encoder ϕ_θ specifies a rate-distortion region for a given task T_k. Moreover, the encoder-prototype-decoder pair ([2](#)) is the rate-distortion function that achieves the infimum rate for a given distortion threshold R_0. The Blahut-Arimoto algorithm ([Arimoto, 1972](#)) was proposed for calculating the rate-distortion function, which is an alternating minimization procedure. This algorithm can be specialized in our setting to pretrain the model to ensure achievability by alternating between encoder and decoder to minimize the empirical risk. In addition, such an achievability constraint in effect imposes an information bottleneck ([Tishby et al., 2000](#)) to balance the compression-representation trade-off.*

With the prototype decoder effectively capturing task-specific characteristics, we next introduce a *Prototype-Decoder Based Meta-Update* method to fine-tune the encoder. This approach prevents overfitting on the training tasks, ensuring that the encoder remains general enough for diverse tasks while preserving the EPD model's ability to adapt effectively to specific tasks online.

3.4. Encoder Meta-Update based on Prototype Decoder

The prototype decoder is a *local* definition that only represents its corresponding task. The *global* relationships among prototypes are embedded within the encoder in a black-box manner, which determines our ability to understand the underlying task similarities and leverage them for task generalization. To explore the global relationships among the prototypes, we introduce an N-dimensional statistical model with the prototype set as a basis in the "task" distribution space (see Figure [2](#)):

Figure 2: Illustration of the statistical model of task distribution.

$$\mathcal{S}_{\mathbf{W}}(\mathbf{a}) = \{\sum_{i=1}^{N} a_i \mathbf{w}_i \mid \sum_{i=1}^{N} a_i = 1 \text{ and } a_i \geq 0\}, \tag{7}$$

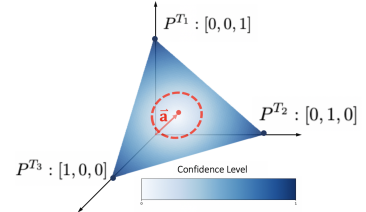where $\mathbf{a} = [a_1, \, a_2, \, \ldots, a_N]^T$ is the *coordinates* in the prototype basis, representing the *location* of a task distribution in this model. With this model structure (7), we introduce a prototype-decoder-based meta-update strategy for jointly training the decoders, focusing on exploring the subspace spanned by the prototype basis. The exploration is achieved by adjusting the learning direction through **negative weighting** of the risk gradients of other tasks' prototypes (see Figure 3). For task $T_k \in \mathcal{T}$, the one-step meta update is given by:

$$\theta \leftarrow \theta - \epsilon \Big((1 - \beta) \nabla_\theta \mathcal{R}_n^{T_k}(\mathbf{w}_k, \phi_\theta) - \beta \sum_{\mathbf{w}' \in \mathbf{W} \backslash \{\mathbf{w}_k\}} \nabla_\theta \mathcal{R}_n^{T_k}(\mathbf{w}', \phi_\theta) \Big), \qquad (8)$$

where $\beta \in [0, 1)$ is a trade-off parameter, balancing task-specific learning and inter-task interpolation, and ϵ is the learning rate. In the case of $\beta = 0$, the task-specific prototype remains highly representative of its corresponding task, yet this choice restricts the interpolation on the statistical model (7). Increasing the value of β broadens the model's interpolation and coverage during the learning phase but will degrade the representativeness in the sense of a higher risk for the given task. The selection of β should align with specific performance metrics: a smaller β for concentrated representation to trained tasks and a larger β for better extrapolation to new tasks.

The meta update is reminiscent of *gradient manipulation* in the multi-task learning (Maninis et al., 2019; Liu et al., 2021a,b), which aims to balance the learning quality between task-shared and task-specific representations. Note that this balance is explicitly addressed by our EPD structure. Here, the use of adversarial gradient regularization is specifically designed to explore the vicinity of a given task by introducing tendencies towards other tasks. The complete algorithm, including the pretraining step, is detailed in the algorithm section of the Appendix in (Gu et al., 2024).
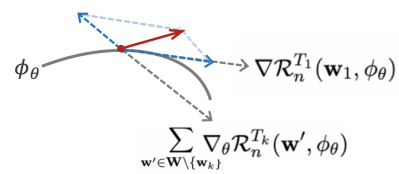

Figure 3: Illustration of meta update. Blue indicates the weighted gradients and red indicates the update direction

3.5. *Proto*-MPC

The EPD model offers an adaptation strategy when used with the MPC to handle uncertainties or disturbances associated with tasks. If privileged information about a task is available online, then *Proto*-MPC can utilize a task-specific residual dynamics model provided by the prototype decoder. Otherwise, in scenarios where task information is not immediately available, we can use prototype decoders to interpolate online data to infer a residual dynamics model.

With Privileged Task Information: Under this condition, MPC can readily choose which model (i.e., prototype decoder) to use. Formally, we describe the task information to be provided by external modules in terms of *Privileged Information* denoted as $\mathtt{PI}$ as follows $\mathbf{w}_k = \mathtt{PI}(D_n^{T_{\text{query}}})$, where $D_n^{T_{\text{query}}} = \{(x_1, y_1), ..., (x_n, y_n)\}^{T_{\text{query}}}$ is a batch of data from the real-time task T_{query}. This operation essentially outputs the prototype candidate to be used by the MPC.

Without Privileged Task Information: When task information is not immediately available during operation, the statistical model $\mathcal{S}_{\mathbf{W}}(\mathbf{a})$ (with prototype decoders as a basis) enables a more efficient sampling-based real-time adaptation strategy than recursively solving the empirical risk mini-

mization. Intuitively, this strategy sequentially *locates* the operational task T_{query} in the (sub)space spanned by the prototype decoders.

Different from the offline learning stage, we shift our focus from exploration to exploitation at the stage of online adaptation. For exploitation, a challenge comes from the center region in $\mathcal{S}_{\mathbf{W}}(\mathbf{a})$ (see Figure 2) being a *low-confidence* region which is poorly represented in the training data. In particular, the point $\mathbf{a}^* = [\frac{1}{N} \ \frac{1}{N} \ ... \ \frac{1}{N}]$ at the center of $\mathcal{S}_{\mathbf{W}}(\mathbf{a})$ represents the state of highest uncertainty, where each task is equally probable. To address this challenge, we propose a **prototype-based coordinates sampling method with an acceptance criterion**, which sequentially updates $\mathbf{a}$ in the high confidence region of $\mathcal{S}_{\mathbf{W}}(\mathbf{a})$.

For the prototype coordinate $\mathbf{a}$, its kth element $\mathbf{a}_k$ has a probabilistic interpretation as the probability of the task T_{query} being T_k, i.e., $\mathbf{a}_k = P(T_{\text{query}} = T_k)$. Therefore, the coordinate $\mathbf{a}$ essentially gives the probability distribution of T_{query} over task set $\mathcal{T}$. In practice, given $D_n^{T_{\text{query}}}$, we can empirically approximate $\mathbf{a}_k$ using Boltzmann distribution:

$$\mathbf{a}_k = P(T_{\text{query}} = T_k) \approx P_{emp}(T_{\text{query}} = T_k) = \mathbf{a}_{emp,k} = \frac{\exp\big(-\gamma \mathcal{R}_n^{T_{\text{query}}}(\mathbf{w}_k, \phi_\theta)\big)}{\sum_{\mathbf{w}' \in \mathbf{W}} \exp\big(-\gamma \mathcal{R}_n^{T_{\text{query}}}(\mathbf{w}', \phi_\theta)\big)}, \quad (9)$$

where $\gamma > 0$ is a scaling parameter that controls the weighting to the risk (i.e., a lower value of γ tends to "flatten out" P_{emp}). To keep $\mathbf{a}_{emp}$ away from the highest uncertain point $\mathbf{a}^*$, we define an *acceptance* criterion using Kullback–Leibler divergence with a predefined acceptance threshold D_0, i.e., if the following inequality holds $D_{KL}(\mathbf{a}_{emp} \| \mathbf{a}^*) > D_0$, then $\mathbf{a}_{emp}$ is considered as bounded away from $\mathbf{a}^*$ and will be accepted. In the inference stage, $\mathbf{a}_{emp}$ can be recursively computed using a *moving horizon data buffer* to sequentially update the decoder weights online while the acceptance criterion ensures that $\mathbf{a}_{emp}$ stays away from the central low-confidence region. The complete algorithm of *Proto*-MPC is detailed in the algorithm section of the Appendix in (Gu et al., 2024). The block diagram of *Proto*-MPC for controlling the quadrotor is illustrated in Fig. 1d.

4. Experiments

In this section, we evaluate our method in simulation. We use the RotorPy simulator (Folk et al., 2023), a multirotor simulation environment with aerodynamic wrenches, to collect data for training the EPD model and test the *Proto*-MPC. Further details on the experimental setup and results can be found in Sections 6.3 and 6.6 of the Appendix in (Gu et al., 2024). Additionally, the learning results of the EPD model within this experimental setup, including the effects of the trade-off parameter β, are discussed in detail in Sections 6.4 and 6.5 of the Appendix in (Gu et al., 2024).

Experimental Setup: The learning task set is designed for constant side wind in the x-direction at speeds of 2, 4, and 6 m/s. In this scenario, the lumped forces dominate the residual dynamics f_Δ. Therefore, only the lumped forces are considered in the learned residual dynamics of this experimental setup.

Experimental Results: To evaluate our method, we compare it with 1) nonlinear MPC with nominal model f_{nom}, 2) KNODE-MPC-Online (Jiahao et al., 2023), and 3) MPC with task-specific DNN residual model ($f_\theta^{T_k}$ is a DNN trained using the T_k-specific dataset). In other words, for each task, a DNN residual model is trained and used for deployment on the given task. On the contrary, for

all the testing trials we conduct in this subsection, the prototype-based meta-model is **kept fixed** so that the adaptation is only based on prototype decoders corresponding to constant side wind with speeds of 2, 4, and 6 m/s. We evaluate our method under static and dynamic wind scenarios. For the former, we command the quadrotors to track the training trajectory under constant side wind of various speeds. For the latter, we command the quadrotors to track different testing trajectories under spatially dependent winds (0–10 m/s along the x-direction; see the illustration in Figure 4).

Constant Side Wind: Table 1 presents a comparison of tracking RMSE for nominal MPC, task-specific DNN-MPC, KNODE-MPC-Online, and *Proto*-MPC under constant side wind conditions with speeds ranging from 0 to 10 m/s. We followed the implementation of KNODE-MPC-Online as described in (Jiahao et al., 2023) for handling sudden mass changes online but adapted it for our side wind setup. Empirically, we found that the original implementation suffers from instability issues with the online learned model in our experimental setup. To address this, we applied spectral normalization to control the Lipschitz constant of the online-learned KNODE model, thereby improving its closed-loop stability.

Figure 4: Spatially varying wind distribution.

The result shows a substantial reduction in RMSE for all task-specific DNN-MPC, KNODE-MPC-Online and *Proto*-MPC compared to the baseline MPC. Note that the task-specific DNN-MPC is expected to exhibit superior tracking performance, as the DNN is specifically trained for each task's wind condition. Both KNODE-MPC-Online and *Proto*-MPC consistently halve the RMSE relative to the nominal MPC across all test wind speeds. However, *Proto*-MPC requires less online computation compared to KNODE-MPC-Online, as it only updates the decoders instead of training the whole model online. This comparison demonstrates not only a significant improvement over the baseline MPC but also demonstrates *Proto*-MPC's robust generalization capabilities on tasks unseen during training, with significantly lower computational demands.

Spatially Varying Wind: Under this condition, to test *Proto*-MPC's task-adaptation capacity, the quadrotor is subject to a varying-speed wind in the x-direction from 0 to 10 m/s. We compare it with nominal-MPC and KNODE-MPC-Online (with spectral normalization) on various trajectories. Figure 5 shows the tracking performance with the colorbar highlighting the deviation from the reference trajectory. Table 2 shows the RMSE of the three methods on the testing trajectories. Compared with nominal MPC and KNODE-MPC-online, the *Proto*-MPC achieves the best trajectory tracking under drastically changing wind conditions with significantly less online computation.

5. Conclusion

This paper proposes a novel EPD model designed to capture shared and distinctive features across various training tasks. The EPD model consists of a universal task-agnostic DNN encoder and a set of task-specific linear prototype decoders to balance task-shared and task-specific representations. In the online setting, the encoder processes incoming data into features. Simultaneously, the linear prototype decoders are used as a "basis" to interpolate encoded features, which allows fast computation of a new decoder aligned with the current task's characteristics. We then use the
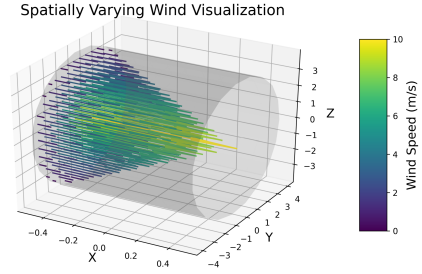
Table 1: Tracking RMSE on the training trajectory under constant side winds of different speeds. The bold font for 2, 4, and 6 m/s cases indicate the wind speeds for the training tasks.

RMSE[m]	axis	0 m/s	**2 m/s**	**4 m/s**	**6 m/s**	8 m/s	10 m/s
nominal-MPC	x	0.10	0.15	0.24	0.36	0.48	0.63
	y	0.07	0.07	0.08	0.08	0.09	0.10
	z	0.03	0.03	0.05	0.08	0.12	0.16
Task-DNN-MPC	x	-	0.08	0.09	0.11	0.12	0.15
	y	-	0.06	0.05	0.06	0.05	0.05
	z	-	0.03	0.03	0.03	0.04	0.04
KNODE-MPC-Online (with Spectral Normalization)	x	0.08	0.09	0.11	0.18	0.26	0.31
	y	0.05	0.07	0.07	0.13	0.16	0.11
	z	0.06	0.05	0.08	0.17	0.18	0.22
Proto-MPC (with `PI`)	x	0.09	0.07	0.09	0.11	0.17	0.30
	y	0.04	0.04	0.05	0.05	0.05	0.05
	z	0.03	0.02	0.03	0.03	0.03	0.04
Proto-MPC (without `PI`)	x	0.10	0.07	0.13	0.17	0.24	0.32
	y	0.04	0.04	0.04	0.05	0.05	0.05
	z	0.03	0.02	0.02	0.03	0.03	0.04

Table 2: Tracking RMSE on the testing trajectories (shown in Figure 5) under spatially varying wind.

RMSE[m]	axis	trajectory 1	trajectory 2	trajectory 3
nominal-MPC	x	0.25	0.31	0.35
	y	0.05	0.06	0.11
	z	0.06	0.06	0.09
KNODE-MPC-Online (with Spectral Normalization)	x	0.15	0.17	0.22
	y	0.06	0.05	0.09
	z	0.09	0.08	0.06
Proto-MPC (without `PI`)	x	0.12	0.15	0.18
	y	0.03	0.03	0.12
	z	0.02	0.02	0.08

EPD model to capture residual dynamics in our *Proto*-MPC, which can quickly adapt the model to cope with uncertainties from dynamically evolving task scenarios. We evaluate *Proto*-MPC's performance in controlling a quadrotor to track agile trajectories under various static and dynamic side wind conditions, which demonstrates its robust performance compared to nominal MPC and its generalization capacity compared to MPC augmented with task-specific DNN residual models. Future directions include deploying this framework in real-world experiments and investigate how the geometric properties of prototype decoders help to better understand the underlying relationships between tasks on the manifold.
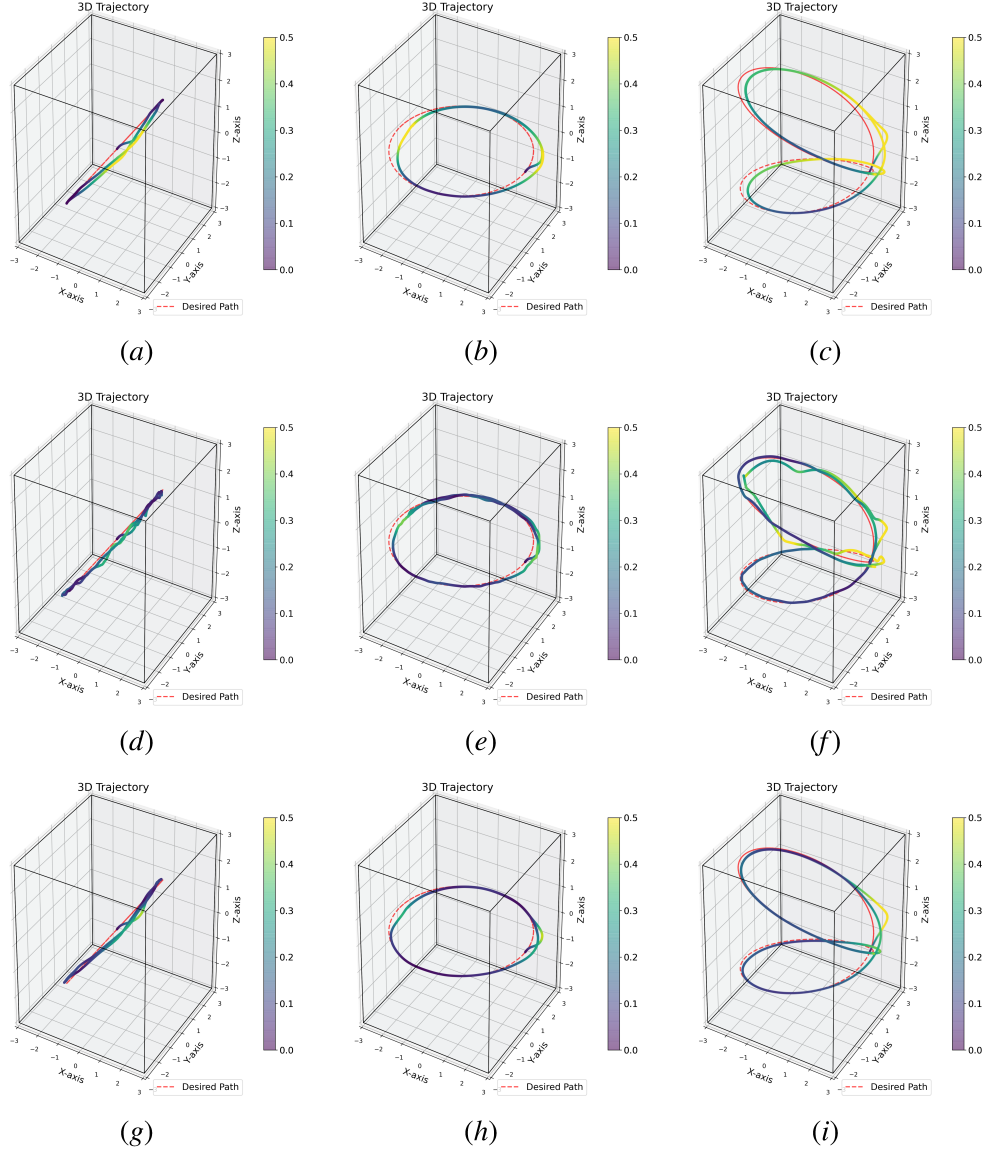
Figure 5: Tracking performance subject to spatially varying winds on different trajectories. The first row (5(*a*), 5(*b*), 5(*c*)) shows the tracking performance of MPC with the nominal model, the second row (5(*d*), 5(*e*), 5(*f*)) shows the tracking performance of KNODE-MPC-Online (with spectral normalization) and the third row (5(*g*), 5(*h*), 5(*i*)) shows the tracking performance of *Proto*-MPC. The colorbar highlights the deviation from the reference trajectory

Acknowledgments

References

Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

Kong Yao Chee, Tom Z Jiahao, and M Ani Hsieh. KNODE-MPC: A knowledge-based data-driven predictive control framework for aerial robots. *IEEE Robotics and Automation Letters*, 7(2): 2819–2826, 2022.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Moritz Diehl, Hans Georg Bock, Holger Diedam, and P-B Wieber. Fast direct multiple shooting algorithms for optimal robot control. *Fast motions in biomechanics and robotics: optimization and feedback control*, pages 65–93, 2006.

Spencer Folk, James Paulos, and Vijay Kumar. RotorPy: A python-based multirotor simulator with aerodynamics for education and research. *arXiv preprint arXiv:2306.04485*, 2023.

Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, 2017.

Tom Z Jiahao, Kong Yao Chee, and M Ani Hsieh. Online dynamics learning for predictive control with an application to aerial robots. In *Proceedings of the Conference on Robot Learning*, pages 2251–2261. PMLR, 2023.

Girish Joshi, Jasvir Virdi, and Girish Chowdhary. Asynchronous deep model reference adaptive control. In *Proceedings of the Conference on Robot Learning*, pages 984–1000. PMLR, 2021.

Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

Nathan O Lambert, Daniel S Drew, Joseph Yaconelli, Sergey Levine, Roberto Calandra, and Kristofer SJ Pister. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robotics and Automation Letters*, 4(4):4224–4230, 2019.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021a.

Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *Proceedings of the International Conference on Learning Representations*, 2021b.

Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1851–1860, 2019.

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525. IEEE, 2011.

Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, 2022.

SM Richards, N Azizan, J-JE Slotine, and M Pavone. Adaptive-control-oriented meta-learning for nonlinear systems. In *Robotics science and systems*, 2021.

Alessandro Saviolo, Guanrui Li, and Giuseppe Loianno. Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking. *IEEE Robotics and Automation Letters*, 7(4):10256–10263, 2022.

Alessandro Saviolo, Jonathan Frey, Abhishek Rathod, Moritz Diehl, and Giuseppe Loianno. Active learning of discrete-time dynamics for uncertainty-aware model predictive control. *IEEE Transactions on Robotics*, 2023.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Guillem Torrente, Elia Kaufmann, Philipp Föhn, and Davide Scaramuzza. Data-driven MPC for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776, 2021.

Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Niels van Duijkeren, Andrea Zanelli, Rien Quirynen, and Moritz Diehl. Towards a modular software package for embedded optimization. *IFAC-PapersOnLine*, 51(20):374–380, 2018.

Bingheng Wang, Zhengtian Ma, Shupeng Lai, and Lin Zhao. Neural moving horizon estimation for robust flight control. *IEEE Transactions on Robotics*, 40:639–659, 2024.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

Zhuohuan Wu, Sheng Cheng, Kasey A Ackerman, Aditya Gahlawat, Arun Lakshmanan, Pan Zhao, and Naira Hovakimyan. $\mathcal{L}_1$ adaptive augmentation for geometric tracking control of quadrotors. In *Proceedings of the 2022 International Conference on Robotics and Automation (ICRA)*, pages 1329–1336. IEEE, 2022.