Model-Based Meta-Reinforcement Learning for Flight with Suspended Payloads

Suneel Belkhale†, Rachel Li†, Gregory Kahn†, Rowan McAllister†, Roberto Calandra‡, Sergey Levine†
†Berkeley AI Research, ‡Facebook AI Research

Abstract—Transporting suspended payloads is challenging for autonomous aerial vehicles because the payload can cause significant and unpredictable changes to the robot’s dynamics. These changes can lead to suboptimal flight performance or even catastrophic failure. Although adaptive control and learning-based methods can in principle adapt to changes in these hybrid robot-payload systems, rapid mid-flight adaptation to payloads that have a priori unknown physical properties remains an open problem. We propose a meta-learning approach that “learns how to learn” models of altered dynamics within seconds of post-connection flight data. Our experiments demonstrate that our online adaptation approach outperforms non-adaptive methods on a series of challenging suspended payload transportation tasks. Videos and other supplemental material are available on our website https://sites.google.com/view/meta-rl-for-flight

I. INTRODUCTION

System identification for optimal control or trajectory optimization is a powerful tool to control well-characterized robotic systems, such as quadcopters, in the absence of any physical interaction with the environment. However, while characterizing the dynamics of an isolated robotic system only needs to be done once per robot, characterizing the physical properties of every possible object in advance is infeasible in open-world environments. Therefore, when these robotic systems physically interact with the world, for example by moving objects, the complexity and unpredictability of these interactions can render manually designed dynamics models insufficient for online control. Unfortunately, it is precisely these physical interactions that constitute the primary modality by which robots influence the world around them, and therefore for a robotic system to affect its environment, it must be able to operate given unknown and unpredictable environmental conditions, object properties, and other physical phenomena.

Learning is another powerful tool for control, in which dynamics models or controllers are trained directly from observed data without relying on domain experts for system identification. However, training these models often requires an exceedingly large amount of data. Data inefficiency poses a significant challenge because many of the ways in which a robot would interact physically with the environment, such as a quadcopter picking up a payload, require fast adaptation. To address the challenge of fast, online adaptation, we propose an approach based on meta-learning. In our meta-learning formulation, the objective we optimize explicitly trains the model for fast adaptation in the case of changing dynamics. We study our approach in the context of a suspended payload control task, in which a quadcopter must position itself to pick up the desired payload with its suspended cable, and transport the payload along a desired path to a goal destination.

An example illustration of such a suspended payload control task is shown in Fig. 1. Although this task is challenging for many reasons—including nonlinear stochastic dynamics—one of the biggest challenges stems from the variability in dynamics induced by different payloads. For example, a payload attached with a shorter cable will oscillate faster compared to one attached with a longer cable. Because the robot will be picking up and dropping off different a priori unknown payloads, the robot must adapt to the new dynamics quickly to avoid instabilities in order to successfully control the payload.

To address these challenges, we present a meta-learning algorithm that enables a quadcopter to adapt to various payloads in an online fashion. Our algorithm can be viewed as a model-based meta-reinforcement learning method: we learn a predictive dynamics model, represented by a deep neural network, which is augmented with stochastic latent variables that represent the unknown factors of variation in the environ-
ment and task. The model is trained with data from a variety of physical conditions, such as different payload masses and tether lengths, using variational inference to estimate the corresponding posterior distribution over these latent variables. This training procedure enables the model to adapt to new payloads at test-time by inferring the posterior distribution over the latent variables. While continuously adapting the model online, a controller uses the model to control the suspended payload along a specified path.

In our experiments, we demonstrate that adaptation is required for accurate quadcopter suspended payload transportation. Our model-based meta-reinforcement learning method enables the quadcopter to perform visual servoing of payloads: the quadcopter uses the model to plan trajectories that will follow desired payload trajectories in the field of view of an external camera, drop off these payloads at designated locations, and even pick up new payloads with a magnetic gripper. We believe this is the first meta-learning approach demonstrated on a real-world quadcopter using only real-world training data that successfully shows improvement in closed-loop performance compared to non-adaptive methods for suspended payload transportation.

II. RELATED WORK

Prior work on control for aerial vehicles has demonstrated impressive performance and agility, such as enabling aerial vehicles to navigate between small openings [18], perform aerobatics [16], and avoid obstacles [24]. These approaches have also enabled aerial vehicles to aggressively control suspended payloads [28], [29]. These methods typically rely on manual system identification, in which the equations of motion are derived and the physical parameters are estimated for both the aerial vehicle [17], [32] and the suspended payload [28], [29]. Although these approaches have successfully enabled controlled flight of the hybrid system, they require a priori knowledge of the system, such as the payload mass and tether length [6]. When such parameters cannot be identified in advance, alternative techniques are required.

Many approaches overcome the limitations of manual system identification by performing automated system identification, in which certain parameters are automatically adapted online according to a specified error metric [27], [11]. However, the principal drawback of manual system identification—the reliance on domain knowledge for the equations of motion—still remains. While certain rigid-body robotic systems are easily identified, more complex phenomena, such as friction, contacts, deformations, and turbulence, may have no known analytic equations (or known solutions). In such cases, data-driven approaches that automatically model a system’s dynamics from data can be advantageous.

Prior work has also proposed end-to-end learning-based approaches that learn from raw data, such as value-based methods which estimate cumulative rewards [30] or policy gradient methods that directly learn a control policy [31]. Although these model-free approaches have been used to learn policies for various tasks [19], [26], including for robots [13], the learning process generally takes hours or even days, making it poorly suited for safety-critical and resource-constrained quadcopters.

Model-based reinforcement learning (MBRL) can provide better sample efficiency, while retaining the benefits of end-to-end learning [4], [8], [20], [4]. With these methods, a dynamics model is learned from data and then used by either a model-based controller or to train a control policy. Although MBRL has successfully learned to control complex systems such as quadcopters [11], [15], most MBRL methods are designed to model a single task with unchanging dynamics, and therefore do not adapt to rapid online changes in the dynamics of a system.

One approach to enable rapid adaptation to time-varying dynamical systems is meta-learning, which is a framework for learning how to learn that typically involves fine-tuning of a model’s parameters [7], [10], [21] or input variables [23], [25]. There has been prior work on model-based meta-learning for quadcopters. O’Connell et al. [22] used the MAML [7] algorithm for adapting a drone’s internal dynamics model in the presence of wind. Although they demonstrated the meta-learning algorithm improved the model’s accuracy, the resulting adapted model did not improve the performance of the closed-loop controller. In contrast, we demonstrate that our meta-learning approach does improve performance of the model-based controller. Nagabandi et al. [21] also explored meta-learning for online adaptation in MBRL for a legged robot, demonstrating improved closed-loop controller performance with the adapted model. Our work focuses on suspended payload manipulation with quadcopters, which presents an especially prominent challenge due to the need for rapid adaptation in order to cope with sudden dynamics changes when picking up payloads.

III. PRELIMINARIES

We first introduce our notation, problem formulation, and preliminaries on model-based reinforcement learning (MBRL) that our meta-learning algorithm builds upon. We represent the hybrid robot-environment system as a Markov decision process, with continuous robot-environment state \( s \in \mathbb{R}^{d_s} \), continuous robot action \( a \in \mathbb{R}^{d_a} \), and discrete time steps \( t \). The state evolves each time step according to an unknown stochastic function \( s_{t+1} \sim p(s_{t+1}|s_t, a_t) \). We consider \( K \) tasks \( \{T_1, ..., T_K\} \). In each task, the robot’s objective is to execute actions that maximize the expected sum of future rewards \( r(s_t, a_t) \in \mathbb{R} \) over the task’s finite time horizon \( T \).

We approach this problem using the framework of model-based reinforcement learning, which estimates the underlying dynamics from data, with minimal prior knowledge of the dynamics of the system. We can train a dynamics model \( p_{\theta}(s_{t+1}|s_t, a_t) \) with parameters \( \theta \) by collecting data in the real world, which we can view as sampling “ground truth” tuples \( (s_t, a_t, s_{t+1}) \). By collecting a sufficient amount of empirical data \( D^{train} = \{(s_0, a_0, s_1), (s_1, a_1, s_2), \ldots\} \), we can train the
parameters $\theta$ of the dynamics model via maximum likelihood

$$
\theta^* = \arg \max_{\theta} p(D^\text{train} | \theta)
= \arg \max_{\theta} \sum_{(s_t, a_t, s_{t+1}) \in D^\text{train}} \log p_{\theta}(s_{t+1} | s_t, a_t).
$$

(1)

To instantiate this method, we extend the PETS algorithm [4], which has previously been shown to handle expressive neural network dynamics models and attain good sample efficiency and final performance. PETS uses an ensemble of neural network models, each parameterizing a Gaussian distribution on $s_{t+1}$ conditioned on both $s_t$ and $a_t$. The learned dynamics model is used to plan and execute actions via model predictive control (MPC) [9, 14, 20]. MPC uses the dynamics model to predict into the future, and selects the action sequence that has the highest predicted reward:

$$
a_t^* = \arg \max_{a_t} \left[ \max_{s_t \sim \mathcal{D}_{a_t}} \sum_{t=1}^{t+H} \mathbb{E}_{s_{t+1} \sim p_{\theta}(s_{t+1} | s_t, a_t)} r(s_t, a_t) \right],
$$

(2)

in which $s_t$ is recursively sampled from the learned dynamics model: $s_{t+1} \sim p_{\theta}(s_{t+1} | s_t, a_t)$, initialized at $s_t \leftarrow s_0$. Once this optimization is solved, only the first action $a_0^*$ is executed. A summary of this MBRL framework is provided in Algorithm 1 and we refer the reader to Chua et al. [4] for additional details.

Algorithm 1 Model-Based Reinforcement Learning

1: Initialize dynamics model $p_{\theta}$ with random parameters $\theta$
2: while not done do
3:   Get current state $s_t$
4:   Solve for action $a_t^*$ given $p_{\theta}$ and $s_t$ using MPC \(\triangleright\) see (2)
5:   Execute action $a_t^*$
6:   Record outcome: $D^\text{train} \leftarrow D^\text{train} \cup \{(s_t, a_t^*, s_{t+1})\}$
7:   Train dynamics model $p_{\theta}$ using $D^\text{train} \triangleleft$ see (1)
8: end while

IV. MODEL-BASED META-LEARNING FOR QUADCOPTER PAYLOAD TRANSPORT

Our goal is to enable a quadcopter to transport various suspended payloads with unknown properties along specified trajectories. The primary challenge is that this interaction is difficult to model a priori because each suspended payload has potentially different physical properties. Although prior work on MBRL has been able to learn to control complex systems, MBRL is unable to account for factors of variation that are not accounted for in the state $s$, such as the unknown properties of a suspended payload of a quadcopter. We approach this problem of accounting for a priori unspecified factors of variation through the lens of meta-learning. Although we only consider the specific task of quadcopter payload transportation in this work, we note that our method is general and is applicable to any robotic system that interact with the environment under changing conditions.

The quadcopter’s objective is to pick up and transport a suspended payload along a specified path to reach a goal location (Fig. 1). First, the quadcopter must fly to the location of the payload (Fig. 1), attach itself to the payload using a suspended cable (Fig. 1b), and then lift the payload off the ground (Fig. 1c). The magnetic gripper is at the end of a tether, so its dynamics are themselves complex and assumed to be unknown before training. As soon as the quadcopter takes off with the payload, the quadcopter’s dynamics change drastically, and therefore online adaption is critical. As the quadcopter flies with the payload towards the goal location (Fig. 1d), our method continuously adapts to the new payload by updating and improving its dynamics model estimate in real time. The adaptive model improves the performance of the MPC controller, which enables the quadcopter to reach the goal destination and release the payload (Fig. 1e). The quadcopter is then able to continue transporting other payloads by adapting online to each new payload it transports.

This section proceeds following our Fig 2 method summary from left to right. First, we detail our data collection method (IV-A). We then describe two variants of our meta-learning method: one in which the properties of the training-time payloads are only known at training time and adaptation is required at test-time (IV-B), and another in which the properties of the payloads are never known (IV-C). Finally, we describe how this method can be used to manipulate payloads at test-time deployment (IV-D), and provide a full algorithmic summary (IV-F).

A. Data Collection

Our method begins by collecting training data (seen Fig. 2 left). We collect data using a person to pilot the quadcopter along random paths for each of the $K$ different suspended payloads, though an automated data collection protocol could also be used instead. We save all the data into a single dataset $D^\text{train}$ consisting of $K$ separate datasets $D^\text{train}_k \in \{D^\text{train}_1, ..., D^\text{train}_K\}$, one per payload task. The quadcopter we use is the DJI Tello (Fig. 1). The Tello is ideal for easy and rapid experimentation for suspended payload control thanks to its small $98 \text{ mm} \times 93 \text{ mm} \times 41 \text{ mm}$ size, light $80 \text{ g}$ weight, long 13 minute battery life, and powerful motors. During different tasks, 3D printed payloads weighing between 10 and 15 grams are attached to the Tello via strings between 18 and 30 centimeters long.

During data collection, we record the controls (actions) and the location of the payload, which we track with an externally mounted RGB camera using OpenCV [3]. The recorded actions are Cartesian velocity commands $\mathbf{a} \in \mathbb{R}^3$ and the recorded states are the pixel location and size of the payload $s \in \mathbb{R}^3$, which are saved every 0.25 seconds into the corresponding dataset in $D^\text{train}$. To enable a learned model to reason about how past states and actions affect the future trajectory of the payload, we modify the state to consist of the concatenation of the past 8 states and actions, resulting in the state having dimension $s \in \mathbb{R}^{48}$. We chose to not include any explicit information about the drone in the state representation to avoid any overly burdensome requirements, such as a motion capture system.
In our experiments, the final dataset $D_{train}$ consisted of approximately 16,000 data points (1.1 hours of flight), which were then used by our meta-learning for model-based reinforcement learning algorithm.

### B. Model Training with Known Dynamics Variables

In this section, we consider the case in which we know all the “factors of variation” in the dynamics across tasks, represented explicitly as a “dynamics variable” $z_k \in \mathbb{R}^d_k$ that is known at training time, but unknown at test-time (deployment). For example, we might know that the only source of variation is the tether length $L$, and therefore we can specify $z_k \leftarrow L_k \forall k$ at training time. We can then learn a single dynamics model $p_\theta$ across all tasks by using $z_k$ as an auxiliary input:

$$s_{t+1} \sim p_\theta(s_{t+1}|s_t, a_t, z_k). \quad (3)$$

Having $z_k$ as an auxiliary input is necessary for accurate modelling because the factors of variation that affect the payload’s dynamics, such as the tether length, are not present in the state $s$, which only tracks the position of the tether end point. The combination of both $s_t$ and $z_k$ is more complete representation of the hybrid robot-payload system state, which enables more accurate predictions.

Training is therefore analogous to (1), but with an additional conditioning on $z_{1:K} = [z_1, ..., z_K]$:

$$\theta^* = \arg \max_\theta \log p(D_{train}|z_{1:K}, \theta)$$

$$= \arg \max_\theta \sum_{k=1}^K \sum_{(s_t, a_t, s_{t+1}) \in D_{train}} \log p_\theta(s_{t+1}|s_t, a_t, z_k). \quad (4)$$

The variables in this training process can be summarized in the graphical model shown in Fig. 3a, in which every variable is observed except for the “true” model parameters $\theta$, which we infer approximately as $\theta^*$ using maximum likelihood estimation in (4).

### C. Meta-Training with Latent Dynamics Variables

The formulation in (4) requires knowing the dynamics variables $z_{1:K}$ at training time. This is a significant assumption because not only does it require domain knowledge to identify all possible factors of variation, but also that we can measure each factor at training time.

To remove this assumption, we now present a more general training procedure that infers the dynamics variables $z_{1:K}$ and the model parameters $\theta$ jointly, as shown by Fig. 3b, without needing to know the semantics or values of $z_{1:K}$. We begin by placing a prior over $z_{1:K} \sim p(z_{1:K}) = \mathcal{N}(0, I)$, and then jointly infer the posterior $p(\theta, z_{1:K}|D_{train})$. We refer to this as meta-training, summarized graphically in Fig. 3b and shown in the broader algorithm flow diagram in Fig. 2 (center).

Unfortunately, inferring $p(\theta, z_{1:K}|D_{train})$ exactly is computationally intractable. We therefore approximate this distribution with an approximate—but tractable—variational posterior [12], which we choose to be a Gaussian with diagonal covariance, factored over tasks,

$$q_{\phi_k}(z_k) = \mathcal{N}(\mu_k, \Sigma_k) \approx p(z_k|D_{train}) \quad \forall \ k \in [K], \quad (5)$$

and parameterized by $\phi_k = \{\mu_k, \Sigma_k\}$.

Our meta-learning training objective is to again maximize the likelihood of the full dataset $D_{train} = D_{1:K}$, analogous to Equation (4). The only difference to (4) is that we must (approximately) marginalize out $z_{1:K}$ because it is unknown:
given the trained (fixed) model parameters $\theta^*$, and the resulting MPC planner. Inference is performed by accumulating transitions $(s_t, a_t, s_{t+1})$ into $D^{\text{test}}$, and using this data and the meta-trained model parameters $\theta^*$ to infer the current value of $z^{\text{test}}$ in real time, as seen in the right side of Fig. 2. A summary of the variables involved in the inference task is given by Fig. 3c.

Similarly to [IV-C], exact inference is intractable, and we therefore use a variational approximation for $z^{\text{test}}$:

$$q_{\phi^{\text{test}}}(z^{\text{test}}) = \mathcal{N}(\mu^{\text{test}}, \Sigma^{\text{test}}) \approx p(z^{\text{test}} | D^{\text{test}}),$$

(8)

parameterized by $\phi^{\text{test}} \equiv \{\mu^{\text{test}}, \Sigma^{\text{test}}\}$. Regardless of training regime ([IV-B] or [IV-C]), inferring $z^{\text{test}}$ uses the same procedure outline below.

To infer the relevant effects that our test-time payload is having on our system, we again use variational inference to optimize $\phi^{\text{test}}$ such that the approximate distribution $q_{\phi^{\text{test}}}(z^{\text{test}})$ is close to the true distribution $p(z^{\text{test}} | D^{\text{test}})$, measured by the Kullback-Leibler divergence:

$$\phi^* \doteq \arg \max_{\phi} -KL(q_{\phi}(z^{\text{test}}) || p(z^{\text{test}} | D^{\text{test}}, \theta^*))$$

$$= \arg \max_{\phi} \mathbb{E}_{z^{\text{test}} \sim q_{\phi}} \log p(z^{\text{test}} | D^{\text{test}}, \theta^*) - \log q_{\phi}(z^{\text{test}})$$

$$= \arg \max_{\phi} \mathbb{E}_{z^{\text{test}} \sim q_{\phi}} \log p(D^{\text{test}} | z^{\text{test}}, \theta^*) - \log q_{\phi}(z^{\text{test}})$$

$$+ \log p(z^{\text{test}}) - \log p(D^{\text{test}} | \theta^*)$$

$$= \arg \max_{\phi} \mathbb{E}_{z^{\text{test}} \sim q_{\phi}} \left[ \sum_{(s_t, a_t, s_{t+1}) \in D^{\text{test}}} \log p_{\theta^*}(s_{t+1} | s_t, a_t, z^{\text{test}}) \right]$$

$$- KL(q_{\phi}(z^{\text{test}}) || p(z^{\text{test}})),$$

(9)

Note the objective (9) corresponds to the test-time ELBO of $D^{\text{test}}$, analogous to training-time ELBO of $D^{\text{train}}$ [6]. Thus (9) scores how well $\phi$ describes the new data $D^{\text{test}}$, under our variational constraint that $q$ is assumed to be Gaussian. Since $\theta^*$ was already inferred at training time, we treat it as a constant during this test-time optimization. Equation (9) is tractable to optimize, and therefore at test time we perform gradient descent online in order to learn $\phi^{\text{test}}$ and therefore improve the predictions of our learned dynamics model.

E. Method Implementation

We instantiate the dynamics model as a neural network consisting of four fully-connected hidden layers of size 200 with swish activations. The model was trained using the Adam optimizer with learning rate 0.001. We used 95% of the data for training and 5% as holdout. The model chosen for evaluation was the one which obtained the lowest loss on the holdout data. We adapted code from a PyTorch implementation of PETS [4] found at https://github.com/quanvuong/handful-of-trials-pytorch.

---

**D. Test-Time Task Inference**

At test time, the robot must adapt online to the new task—such as a different type of payload—by inferring the unknown dynamics variables $z^{\text{test}}$ in order to improve the learned dynamics model $p_{\theta^*}$ and the resulting MPC planner. Inference is performed by accumulating transitions $(s_t, a_t, s_{t+1})$ into $D^{\text{test}}$, and using this data and the meta-trained model parameters $\theta^*$ to infer the current value of $z^{\text{test}}$ in real time, as seen in the right side of Fig. 2. A summary of the variables involved in the inference task is given by Fig. 3c.

Similarly to [IV-C], exact inference is intractable, and we therefore use a variational approximation for $z^{\text{test}}$:

$$q_{\phi^{\text{test}}}(z^{\text{test}}) = \mathcal{N}(\mu^{\text{test}}, \Sigma^{\text{test}}) \approx p(z^{\text{test}} | D^{\text{test}}),$$

(8)

parameterized by $\phi^{\text{test}} \equiv \{\mu^{\text{test}}, \Sigma^{\text{test}}\}$. Regardless of training regime ([IV-B] or [IV-C]), inferring $z^{\text{test}}$ uses the same procedure outline below.

To infer the relevant effects that our test-time payload is having on our system, we again use variational inference to optimize $\phi^{\text{test}}$ such that the approximate distribution $q_{\phi^{\text{test}}}(z^{\text{test}})$ is close to the true distribution $p(z^{\text{test}} | D^{\text{test}})$, measured by the Kullback-Leibler divergence:

$$\phi^* \doteq \arg \max_{\phi} -KL(q_{\phi}(z^{\text{test}}) || p(z^{\text{test}} | D^{\text{test}}, \theta^*))$$

$$= \arg \max_{\phi} \mathbb{E}_{z^{\text{test}} \sim q_{\phi}} \log p(z^{\text{test}} | D^{\text{test}}, \theta^*) - \log q_{\phi}(z^{\text{test}})$$

$$= \arg \max_{\phi} \mathbb{E}_{z^{\text{test}} \sim q_{\phi}} \log p(D^{\text{test}} | z^{\text{test}}, \theta^*) - \log q_{\phi}(z^{\text{test}})$$

$$+ \log p(z^{\text{test}}) - \log p(D^{\text{test}} | \theta^*)$$

$$= \arg \max_{\phi} \mathbb{E}_{z^{\text{test}} \sim q_{\phi}} \left[ \sum_{(s_t, a_t, s_{t+1}) \in D^{\text{test}}} \log p_{\theta^*}(s_{t+1} | s_t, a_t, z^{\text{test}}) \right]$$

$$- KL(q_{\phi}(z^{\text{test}}) || p(z^{\text{test}})),$$

(9)

Note the objective (9) corresponds to the test-time ELBO of $D^{\text{test}}$, analogous to training-time ELBO of $D^{\text{train}}$ [6]. Thus (9) scores how well $\phi$ describes the new data $D^{\text{test}}$, under our variational constraint that $q$ is assumed to be Gaussian. Since $\theta^*$ was already inferred at training time, we treat it as a constant during this test-time optimization. Equation (9) is tractable to optimize, and therefore at test time we perform gradient descent online in order to learn $\phi^{\text{test}}$ and therefore improve the predictions of our learned dynamics model.

---

**E. Method Implementation**

We instantiate the dynamics model as a neural network consisting of four fully-connected hidden layers of size 200 with swish activations. The model was trained using the Adam optimizer with learning rate 0.001. We used 95% of the data for training and 5% as holdout. The model chosen for evaluation was the one which obtained the lowest loss on the holdout data. We adapted code from a PyTorch implementation of PETS [4] found at https://github.com/quanvuong/handful-of-trials-pytorch.
Algorithm 2 Model-Based Meta-Reinforcement Learning for Quadcopter Payload Transport

1: // Training Phase
2: For Task $k = 1$ to $K$
3:  For Time $t = 0$ to $T$
4:      Get action $a_t$ from human pilot
5:      Execute action $a_t$
6:      if $z_k$ is known then
7:          Record outcome: $D_{train} \leftarrow D_{train} \cup \{s_t, a_t, s_{t+1}, z_k\}$  // case IV-B
8:      else
9:          Record outcome: $D_{train} \leftarrow D_{train} \cup \{s_t, a_t, s_{t+1}\}$  // case IV-C
10:     end if
11: End for
12: End for
13: Train dynamics model $p_{\theta^*}$ given $D_{train}$ ⊢ see §[IV-B]
14:  
15: // Test Phase
16: Initialize variational parameters: $\phi^* \leftarrow \{\mu^{test} = 0, \Sigma^{test} = I\}$
17: For Time $t = 0$ to $T$
18:      Solve optimal action $a_t^{*}$ given $p_{\theta^*}$, $q_{\phi^*}$, and MPC ⊢ see §[IV-C]
19:      Execute action $a_t^{*}$
20:      Record outcome: $D_{test} \leftarrow D_{test} \cup \{s_t, a_t^{*}, s_{t+1}\}$
21:      Infer variational parameters $\phi^*$ given $D_{test}$ ⊢ see §[IV-C]
22: End for

F. Method Summary

A summary of the full training and test-time procedure is provided in both Fig. 2 and Algorithm 2. During the training phase, a human pilot gathers data for $K$ different tasks consisting of suspended payloads with different dynamics. During flight, tuples $\{s_t, a_t, s_{t+1}\}$ are recorded into the corresponding task dataset, as well as the dynamics variable $z_k$ if it is known (§IV-B). We then train the dynamics model $p_{\theta^*}$ using the dataset $D_{train}$ via (7).

At test time, we initialize $q_{\phi^*}(z^{test})$ to be the prior $\mathcal{N}(0, I)$ and the quadcopter begins to transport payloads with a priori unknown physical properties $z^{test}$. At each time step, we solve for the optimal action $a_t^{*}$ given $p_{\theta^*}$, and the current estimate of $z^{test}$ using the MPC planner in (3). The quadcopter executes the resulting action and records the observed transition $\{s_t, a_t^{*}, s_{t+1}\}$ into the test dataset $D_{test}$. We then adapt the latent variable online by inferring $q_{\phi^*}(z^{test})$ using $D_{test}$. The quadcopter continues to plan, execute, and adapt online until the payload transportation task is complete.

V. EXPERIMENTAL EVALUATION

We now present an experimental evaluation of our meta-learning approach in the context of quadcopter suspended payload control tasks. Videos and supplementary material are available on our website.¹

1: In our experiments, we aim to answer the following questions:

Q1 Does online adaptation via meta-learning lead to better performance compared to non-adaptive methods?

Q2 How does our meta-learning approach compare to MBRL with concatenated past states and actions?

Q3 How does our approach with known versus unknown dynamics variables compare?

Q4 Is our test-time inference procedure able to differentiate between different a priori unknown payloads?

Q5 Can our approach enable a quadcopter to fulfill a complete payload pick-up, transport, and drop-off task?

Q6 What other realistic suspended payload transportation scenarios can our approach enable?

We evaluated our meta-learning approach with both known variables (§IV-B) and latent variables (§IV-C), and compared to multiple other approaches, including:

- **MBRL without history**, in which the state consists of only the current payload pixel location and size.
- **MBRL**, a simple meta-learning approach in which the state consists of the past 8 states and actions concatenated together.
- **PID controller**, which consists of three PID controllers, one for each Cartesian velocity command axis. We manually tuned the PID gains by evaluating the performance of the controller on a trajectory following path not used in our experiments for a single payload, which is representative of how PID controllers are typically tuned.

1 Trajectory Following: We first evaluate the ability of our method to track specified payload trajectories in isolation, separately from the full payload transportation task. Each task consists of following either a circle or square path (Fig. 4) in the image plane or a figure-8 path parallel to the ground, and with a suspended cable length either 18cm or 30cm long. Although the training data included payloads with these cable lengths, the cable length was unknown to all methods during these test-time experiments.

- **Circle**
- **Square**
- **Figure-8**

Fig. 4: Comparison of our meta-learning approach with unknown and known factors of variation versus model-based reinforcement learning (MBRL) with past states and actions concatenated. The tasks are to either follow a circle or square in the image plane, or a figure-8 parallel to the ground. The specified goal paths are colored in red and the path taken by each approach is shown in cyan. Our approaches are better able to adapt online and follow the specified trajectories.

1 https://sites.google.com/view/meta-rl-for-flight
Table I shows the results for each approach in terms of average pixel tracking error, with visualizations of a subset of the executions shown in Fig. 4. Both the online adaptation methods—our approach and MBRL—better track the specified goal trajectories compared to the non-adaptation methods—MBRL without history and PID controller—which shows that online adaptation leads to better performance (Q1). Our meta-learning approach also outperforms the other meta-learning method MBRL (Q2). Our approach meta-trained with unknown latent dynamics variables also outperforms our approach trained with known dynamics models (Q3), which highlights that our approach does not require a priori knowledge of the suspended payloads during training to successfully adapt at test time.

In addition to our method exhibiting better closed-loop performance, the dynamics variable of our model is interpretable. Fig. 5 and Fig. 6 show the inferred dynamics variable and tracking error while our model-based policy is executing at test time. We observe that the dynamics variable converges to different values depending on the cable length, which shows that our test-time inference procedure is able to differentiate between the dynamics of the two different payloads (Q4). More importantly, as the inferred value converges, our learned model-based controller becomes more accurate and is therefore better able to track the desired path (Q1).

2) End-to-End Payload Transportation: We also evaluated our approach on a full end-to-end payload transportation task (Fig. 7), in which the quadcopter must follow a desired trajectory to the payload, attach to the payload using a magnet, lift the payload and transport it along a specified trajectory to the goal location, drop off the payload, and then follow a trajectory back to the start location.

Fig. 7 shows images of the quadcopter (using our approach) executing the full task, along with plots showing the average tracking error and inferred latent value over time. Our approach is able to successfully complete the full task (Q5) due to our online adaptation mechanism (Q1), which enables the drone to better track the specified trajectories and pick up the payload by automatically inferring whether the payload is attached or detached (Q4). Furthermore, the continuous aspect of this demonstration highlights the importance of online adaptation: each time the quadcopter transitions between transporting a payload and not transporting a payload, the quadcopter must re-adapt online to be able to successfully follow the specified trajectories.

3) Additional use cases: In addition to enabling trajectory following and end-to-end payload transportation, our approach can enable a quadcopter to transport a suspended payload (Q6): around an obstacle by following a predefined path (Fig. 5), to greedily follow a target (Fig. 9), and along trajectories dictated using a “wand”-like interface (Fig. 10).
Fig. 6: As the quadcopter follows the circle trajectory using our model-based controller, our approach adapts online to the a priori unknown payload by inferring the latent value which maximizes the dynamics models accuracy. This online adaptation reduces the tracking error as the quadcopter flies, enabling the quadcopter to successfully complete the task.

Fig. 7: Visualization of our approach successfully completing the full quadcopter payload transportation task. The task consists of three distinct phases: before the quadcopter picks up the payload, while the payload is in transit to the goal, and after the payload is dropped off. Our approach continuously adapts the latent dynamics variable online using the current test-time dataset, and flushes the test-time dataset each time the quadcopter transitions between phases, which are delineated by the vertical black lines. The inferred latent variable is the same for when no payload is attached, but different when the payload is attached, which demonstrates that our inference procedure successfully infers the latent variable depending on the payload. Within each phase, the tracking error also reduces over time, which shows that our online adaptation mechanism improves closed-loop performance.

VI. DISCUSSION & CONCLUSION

We presented a meta-learning approach for model-based reinforcement learning that enables a quadcopter to adapt to various payloads in an online fashion. At the core of our approach is a deep neural network dynamics model that learns to predict how the quadcopter’s actions affect the flight path of the payload. We augment this dynamics model with stochastic latent variables, which represent unknown factors of variation in the task. These latent variables are trained to improve the accuracy of the dynamics model and be amenable for fast online adaptation. Our experiments demonstrate that the proposed training and online adaptation mechanisms improve performance for real-world quadcopter suspended payload transportation tasks compared to other adaptation approaches.

Although our approach enabled successful aerial payload transportation for certain tasks, we limited the quadcopter’s action space to only Cartesian velocities. Investigating how to learn to actuate all dimensions of the quadcopter will be important for accomplishing more complex payload transportation tasks. Additionally, our approach assumes an estimate of the suspended payload’s position. Learning directly from raw images could alleviate the effort required to localize the payload, while possibly enabling even better online adaptation. Finally, our approach required manually specifying when the suspended payload was picked up or dropped off. Developing an algorithm that does not require this human oversight would further increase the autonomy of the aerial payload system. We believe that solving these and other challenges is crucial for creating a successful aerial payload transportation platform,
and that our approach is a step towards this goal.

**Contributions**

Suneel implemented the robot and meta-learning algorithm code, created the suspended payload rig, and performed the experiments. Rachel prototyped the meta-learning algorithm and created all the multimedia. Greg created the base quadcopter setup and contributed to write the manuscript. Rowan developed the meta-learning algorithm and contributed to write the manuscript. Roberto provided guidance throughout the project and contributed to write the manuscript. Sergey provided the primary guidance throughout the project and contributed to write the manuscript.

**Acknowledgments**

We thank Simon Ramstedt for recommending the Tello drone and Somil Bansal for providing a quadcopter simulator. This research was supported by the National Science Foundation under IIS-1700697 and IIS-1651843, ARL DCIST CRA W911NF-17-2-0181, NASA ESI, the DARPA Assured Autonomy Program, and the Office of Naval Research, as well as support from Google, NVIDIA, and Amazon.

**References**

[1] Somil Bansal, Anayo K Akametalu, Frank J Jiang, Forrest Laine, and Claire J Tomlin. Learning quadrotor dynamics using neural network for flight control. *IEEE Conference on Decision and Control (CDC)*, pages 4653–4660, 2016.

[2] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[3] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O’Reilly Media, 2008.

[4] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Neural Information Processing Systems (NeurIPS)*, pages 4754–4765, 2018.

[5] Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pages 465–472, 2011.

[6] Aleksandra Faust, Ivana Palunko, Patricio Cruz, Rafael Fierro, and Lydia Tapia. Automated aerial suspended cargo delivery through reinforcement learning. *Artificial Intelligence*, 247:381–398, 2017. ISSN 00043702. doi: 10.1016/j.artint.2014.11.009.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, volume 70, pages 1126–1135, 2017.

[8] Yarin Gal, Rowan McAllister, and Carl Edward Ras-
mussen. Improving PILCO with Bayesian neural network dynamics models. In *ICML Workshop on Data-Efficient Machine Learning*, volume 4, page 34, 2016.

9. Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practicea survey. *Automatica*, 25(3):335–348, 1989.

10. James Harrison, Apoorva Sharma, Roberto Calandra, and Marco Pavone. Control adaptation via meta-learning dynamics. In *NeurIPS Workshop on Meta-Learning*, 2018.

11. Petros Ioannou and Barış Fidan. *Adaptive control tutorial*. SIAM, 2006.

12. Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

13. Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

14. Sanket Kamthe and Marc Peter Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. *arXiv preprint arXiv:1706.06491*, 2017.

15. Nathan O. Lambert, Daniel S. Drew, Joseph Yaconelli, Sergey Levine, Roberto Calandra, and Kristofer S. J. Pister. Low level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):4224–4230, 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2930489.

16. Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D’Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1642–1648, 2010.

17. Robert Mahony, Vijay Kumar, and Peter Corke. Multicopter aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32, 2012.

18. Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *International Journal of Robotics Research*, 31(5):664–674, 2012.

19. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

20. Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566, 2018.

21. Anusha Nagabandi, Ignasi Clavera, Simin Liu, and Ronald S Fearing. Learning to adapt in dynamic, real-world environments via meta-reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

22. Michael O’Connell, Guanya Shi, Xichen Shi, and Soon-Jo Chung. Meta-learning-based robust adaptive flight control under uncertain wind conditions. *Caltech Preprint*, 2019.

23. Christian F Perez, Felipe Petroski Such, and Theofanis Karaletsos. Efficient transfer learning and online adaptation with latent variable models for continuous control. *arXiv preprint arXiv:1812.03399*, 2018.

24. Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer, 2016.

25. Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. Meta reinforcement learning with latent variable Gaussian processes. *arXiv preprint arXiv:1803.07551*, 2018.

26. John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

27. Jean-Jacques E Slotine and Weiping Li. On the adaptive control of robot manipulators. *International Journal of Robotics Research (IJRR)*, 6(3):49–59, 1987.

28. Sarah Tang and Vijay Kumar. Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2216–2222, 2015.

29. Sarah Tang, Valentin Wüest, and Vijay Kumar. Aggressive flight with suspended payloads using vision-based control. *IEEE Robotics and Automation Letters (RA-L)*, 3(2):1152–1159, 2018.

30. Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

31. Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

32. Xiaodong Zhang, Xiaoli Li, Kang Wang, and Yanjun Lu. A survey of modelling and identification of quadrotor robot. In *Abstract and Applied Analysis*, volume 2014. Hindawi, 2014.