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

Suneel Belkhale[1], Rachel Li[1], Gregory Kahn[1], Rowan McAllister[1], Roberto Calandra[2], Sergey Levine[1]


(a) (b) (c) (d) (e)

Fig. 1: Our meta-reinforcement learning method controlling a quadcopter transporting a suspended payload. This task is challenging since each payload induces different system dynamics, which requires the quadcopter controller to adapt online. The controller learned via our meta-learning approach is able to (a) fly towards the payload, (b) attach the cable tether to the payload using a magnet, (c) take off, (d) fly towards the goal location while adapting to the newly attached payload, and (e) deposit the payload using an external detaching mechanism.

***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**

***Index Terms*—Machine Learning for Robot Control, Reinforcement Learning, Probabilistic Inference.**

## I. Introduction

CONSIDER the task illustrated in Figure 1: the quadcopter must maneuver such that the magnet (red) at the end of the tether picks up the payload (green), then lift the payload, and fly such that this payload follows a desired trajectory. While the dynamics of the quadcopter may be well-characterized, and system identification methods could accurately identify its parameters, the complex interaction between the magnetic gripper and the payload are unlikely to be represented accurately by hand-designed models. Even more unpredictable is the effect of the payload on the dynamics of the quadcopter when the payload is lifted off the ground. 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 various *a priori* unknown payloads, the robot must rapidly adapt to the new dynamics to remain in flight. Learning can offer us a powerful tool for handling complex interactions, such as those between the magnetic gripper and the payload. Conventional learning-based methods, however, typically require a large amount of data to learn accurate models, and therefore may be slow to adapt. The payload adaptation task illustrates the need for fast adaptation: the robot must very quickly determine the payload parameters, and then adjust its motor commands accordingly. To address the challenge of rapid *online* adaptation, we propose an approach based on meta-learning. In our meta-learning formulation, we explicitly train the model for fast adaptation to scenarios with new dynamics, such as the task in Figure 1.

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 environment and task. The model is trained with different payload masses and tether lengths, and uses 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. Our novel contribution is in leveraging neural network dynamics models in conjunction with meta-learning for the task of controlling an aerial robot with a suspended payload.

In the experiments, we demonstrate that our method enables a quadcopter to plan and execute trajectories that follow desired payload trajectories, drop off these payloads at des-

Manuscript received: October, 15, 2020; Revised December, 22, 2020; Accepted January, 14, 2021.

This paper was recommended for publication by Editor Dana Kulic upon evaluation of the Associate Editor and Reviewers' comments. 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.

[1]University of California, Berkeley {skbelkhale, rachel_li, gkahn, rmcallister, svlevine}@berkeley.edu

[2]Facebook AI Research. rcalandra@fb.com

Digital Object Identifier (DOI): 10.1109/LRA.2021.3057046.

ignated locations, and even pick up new payloads with a magnetic gripper. To our knowledge, 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 [21], perform aerobatics [19], and avoid obstacles [27]. These approaches have also enabled aerial vehicles to aggressively control suspended payloads [31, 32]. 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 [20, 35] and the suspended payload [31, 32]. 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 [7]. 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 [30, 12, 17]. 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 suspended payload control, adaptive model-based controllers could require *a priori* knowledge of the quadrotor dynamics and suspended payload equations, 3d state estimation of both the quadrotor and suspended payload, and camera parameters in order to perform state estimation. In contrast, our data-driven method only requires the pixel location of the payload and the quadrotor's commanded actions to control and adapt to the suspended payload's dynamics.

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 [33] or policy gradient methods that directly learn a control policy [34]. Although these model-free approaches have been applied to various tasks [22, 29], including for robots [14], 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 [6, 9, 23, 5]. With these methods, a dynamics model is learned from data and then either used by a model-based controller or to train a control policy. Although MBRL has successfully learned to control complex systems such as quadcopters [1, 18], most MBRL methods are designed to model a single task with unchanging dynamics, and therefore do not adapt to rapid online changes in the system dynamics.

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 [8, 11, 24] or input variables [26, 28]. There has been prior work on model-based meta-learning for quadcopters. O'Connell et al. [25] used the MAML [8] 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. [24] and Kaushik et al. [16] 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 $\mathbf{s} \in \mathbb{R}^{d_\mathbf{s}}$, continuous robot action $\mathbf{a} \in \mathbb{R}^{d_\mathbf{a}}$, and discrete time steps $t$. The state evolves each time step according to an unknown stochastic function $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. We consider $K$ tasks $\{\mathcal{T}_1., ..., \mathcal{T}_K\}$. In each task, the robot's objective is to execute actions that maximize the expected sum of future rewards $r(\mathbf{s}_t, \mathbf{a}_t) \in \mathbb{R}$ 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 $p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ with parameters $\theta$ by collecting data in the real world, which we can view as sampling "ground truth" tuples $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. By collecting a sufficient amount of empirical data $\mathcal{D}^{\text{train}} = \{(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1), (\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2), ...\}$, we can train the parameters $\theta$ of the dynamics model via maximum likelihood

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmax}}\ p(\mathcal{D}^{\text{train}}|\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}^{\text{train}}} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\,. \end{aligned} \quad (1)$$

To instantiate this method, we extend the PETS algorithm [5], 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 $\mathbf{s}_{t+1}$ conditioned on both $\mathbf{s}_t$ and $\mathbf{a}_t$. The learned dynamics model is used to plan and execute actions via model predictive control (MPC) [10, 15, 23]. MPC uses the dynamics model to predict into the future, and selects the action sequence with

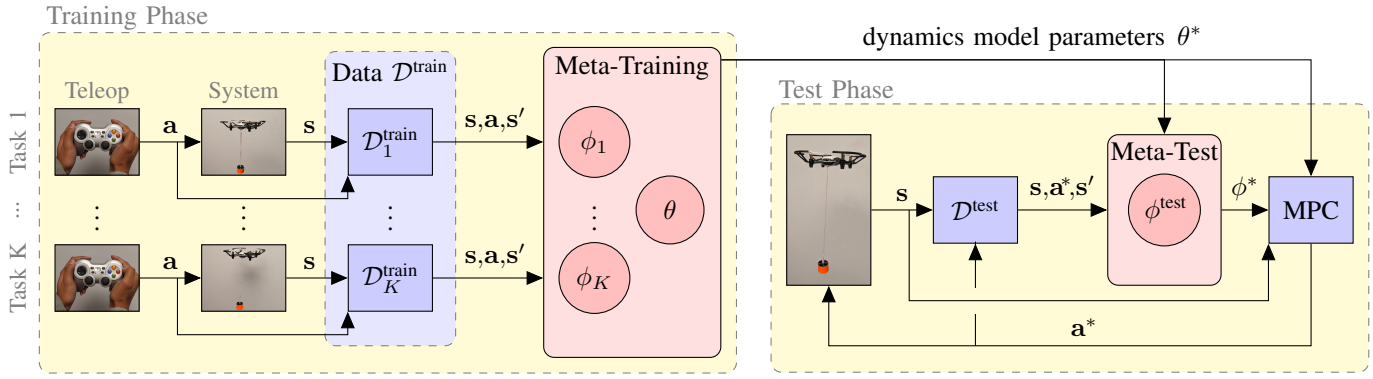



Fig. 3: System diagram of our meta-learning for model-based reinforcement learning algorithm. In the training phase, we first gather data by manually piloting the quadcopter along random trajectories with $K$ different payloads, and saving the data into a single dataset $\mathcal{D}^{\text{train}}$ consisting of $K$ separate training task-specific datasets $\mathcal{D}^{\text{train}} \doteq \mathcal{D}^{\text{train}}_{1:K}$. We then run meta-training to learn the shared dynamics model parameters $\theta$ and the adaptation parameters $\phi_{1:K}$ for each payload task. At test time, using the learned dynamics model parameters $\theta^*$, the robot infers the optimal latent variable $\phi^*$ online using all of the data $\mathcal{D}^{\text{test}}$ from the current task. The dynamics model, parameterized by $\theta^*$ and $\phi^*$, is used by a model-predictive controller (MPC) to plan and execute actions that follow the specified path. As the robot flies, it continues to store data, infer the optimal latent variable parameters, and perform planning in a continuous loop until the task is complete.

the highest predicted reward

$$\mathbf{a}_t^* = \underset{\mathbf{a}_t}{\text{argmax}} \left[ \max_{\mathbf{a}_{t+1:t+H}} \sum_{\tau=t}^{t+H} \mathbb{E}_{\mathbf{s}_\tau \sim p_\theta} \left[ r(\mathbf{s}_\tau, \mathbf{a}_\tau) \right] \right], \quad (2)$$

in which $\mathbf{s}_\tau$ is recursively sampled from the learned dynamics model: $\mathbf{s}_{\tau+1} \sim p_\theta(\mathbf{s}_{\tau+1}|\mathbf{s}_\tau, \mathbf{a}_\tau)$, initialized at $\mathbf{s}_\tau \leftarrow \mathbf{s}_t$. Once this optimization is solved, only the first action $\mathbf{a}_t^*$ is executed. A summary of this MBRL framework is provided in Algorithm 1, and we refer the reader to Chua et al. [5] 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 $\mathbf{s}_t$
4: Solve for action $\mathbf{a}_t^*$ given $p_{\theta^*}$ and $\mathbf{s}_t$ using MPC ▷ see (2)
5: Execute action $\mathbf{a}_t^*$
6: Record outcome: $\mathcal{D}^{\text{train}} \leftarrow \mathcal{D}^{\text{train}} \cup \{\mathbf{s}_t, \mathbf{a}_t^*, \mathbf{s}_{t+1}\}$
7: Train dynamics model $p_\theta$ using $\mathcal{D}^{\text{train}}$ ▷ see (1)

## IV. Model-Based Meta-Learning for Quadcopter Payload Transport

Our goal is to enable a quadcopter to precisely control a wide variety of payloads without prior knowledge of the payload's physical properties. The primary challenge is that the interaction between quadcopter actions and the suspended payload state varies based on the type of payload, and these variations in dynamics are difficult to identify and model *a priori*. 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 $\mathbf{s}$. We approach this problem of accounting for *a priori* unspecified factors of variation through the lens of meta-learning, in which we learn a model that is explicitly trained to adapt online.

The quadcopter's objective is to pick up and transport a suspended payload along a specified path to reach a goal location (Figure 1). First, the quadcopter must fly to the location of the payload (Figure 1a), attach itself to the payload using a suspended cable (Figure 1b), and then lift the payload off the ground (Figure 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 (Figure 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 (Figure 1e). The quadcopter is then able to continue transporting other payloads by adapting online to each new payload it transports.

### A. Data Collection

We first collect data by manually piloting the quadcopter (Figure 3, left) along random paths for each of the $K$ suspended payloads, though any off-policy data collection method that visits a diverse number of state and action sequences could also be used. We save all the data into a single dataset $\mathcal{D}^{\text{train}}$, consisting of $K$ separate datasets $\mathcal{D}^{\text{train}} \doteq \mathcal{D}^{\text{train}}_{1:K} \doteq \{\mathcal{D}^{\text{train}}_1, ..., \mathcal{D}^{\text{train}}_K\}$, one per payload task.

### 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" $\mathbf{z}_k \in \mathbb{R}^{d_\mathbf{z}}$ 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 $\mathbf{z}_k \leftarrow L_k \, \forall \, k$ at training time. We can then learn a single dynamics model $p_\theta$ across all tasks by using $\mathbf{z}_k$ as an auxiliary input to PETS [5]:

$$\mathbf{s}_{t+1} \sim p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_k). \quad (3)$$

Having $\mathbf{z}_k$ 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 $\mathbf{s}$, which only tracks the position of the tether end point. The combination of both $\mathbf{s}_t$ and $\mathbf{z}_t$ 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 $\mathbf{z}_{1:K} \doteq [\mathbf{z}_1, ..., \mathbf{z}_K]$:

$$\begin{aligned}\theta^* &\doteq \underset{\theta}{\operatorname{argmax}} \log p(\mathcal{D}^{\text{train}}|\mathbf{z}_{1:K}, \theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{k=1}^{\text{K}} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}_k^{\text{train}}} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_k)\,. \quad (4)\end{aligned}$$

The variables in this training process can be summarized in the graphical model shown in Figure 4a, 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 §IV-B requires knowing the dynamics variables $\mathbf{z}_{1:K}$ 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 $\mathbf{z}_{1:K}$ and the model parameters $\theta$ *jointly*, as shown by Figure 4b, without needing to know the semantics or values of $\mathbf{z}_{1:K}$. We begin by placing a prior over $\mathbf{z}_{1:K} \sim p(\mathbf{z}_{1:K}) = \mathcal{N}(0, I)$, and then jointly infer the posterior $p(\theta, \mathbf{z}_{1:K}|\mathcal{D}_{1:K}^{\text{train}})$. We refer to this as *meta-training*, summarized graphically in Figure 4b and shown in the broader algorithm flow diagram in Figure 3 (center).

Unfortunately, inferring $p(\theta, \mathbf{z}_{1:K}|\mathcal{D}_{1:K}^{\text{train}})$ exactly is computationally intractable. We therefore approximate this distribution with an approximate—but tractable—variational posterior [13], which we choose to be a Gaussian with diagonal covariance, factored over tasks,

$$q_{\phi_k}(\mathbf{z}_k) \;=\; \mathcal{N}(\mu_k, \Sigma_k) \;\approx\; p(\mathbf{z}_k|\mathcal{D}^{\text{train}}) \quad \forall\, k \in [K]\,, \quad (5)$$

and parameterized by $\phi_k \doteq \{\mu_K, \Sigma_k\}$. Our meta-learning training objective is to again maximize the likelihood of the full dataset $\mathcal{D}^{\text{train}} = \mathcal{D}_{1:K}^{\text{train}}$, analogous to Equation (4). The only difference to §IV-B is that we must (approximately) marginalize out $\mathbf{z}_{1:K}$ because it is unknown:

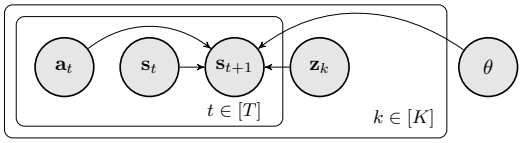
$$\begin{aligned}\log p(\mathcal{D}^{\text{train}}|\theta) \;&=\; \log \int_{\mathbf{z}_{1:K}} p(\mathcal{D}^{\text{train}}|\mathbf{z}_{1:K}, \theta) p(\mathbf{z}_{1:K}) \mathrm{d}\mathbf{z}_{1:K} \\ &= \sum_{k=1}^{K} \log \mathbb{E}_{\mathbf{z}_k \sim q_{\phi_k}} p(\mathcal{D}^{\text{train}}|\mathbf{z}_k, \theta) \cdot p(\mathbf{z}_k)/q_{\phi_k}(\mathbf{z}_k) \\ &\geq \sum_{k=1}^{K} \mathbb{E}_{\mathbf{z}_k \sim q_{\phi_k}} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}_k^{\text{train}}} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_k) - \mathrm{KL}(q_{\phi_k}(\mathbf{z}_k)||p(\mathbf{z}_k)) \\ &\doteq \text{ELBO}(\mathcal{D}^{\text{train}}|\theta, \phi_{1:K})\,. \quad (6)\end{aligned}$$

The evidence lower bound (ELBO) above is a computationally tractable approximate to $\log p(\mathcal{D}^{\text{train}}|\theta)$. For additional details on variational inference, we refer the reader to Bishop [2].

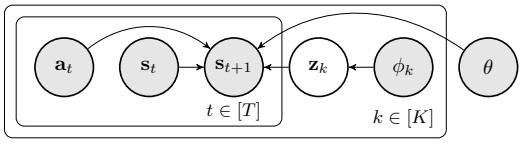
Our meta-training algorithm then optimizes both $\theta$ and the variational parameters $\phi_{1:K}$ of each task with respect to the evidence lower bound

$$\theta^* \;\doteq\; \underset{\theta}{\operatorname{argmax}} \max_{\phi_{1:K}} \text{ELBO}(\mathcal{D}^{\text{train}}|\theta, \phi_{1:K})\,. \quad (7)$$

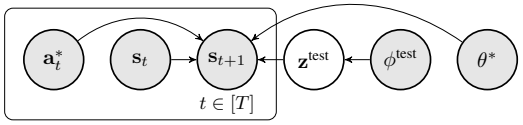
Note that $\theta^*$ will be used at test time, while the learned variational parameters $\phi_{1:K}$ will not be used at test time because the test task can be different from the training tasks.




(a) **Training**-time payloads with **known** factors of variation $\mathbf{z}_k$.




(b) **Training**-time payloads with **unknown** factors of variation $\mathbf{z}_k$.




(c) **Test**-time payload with unknown factors of variation $\mathbf{z}^{\text{test}}$.

Fig. 4: Probabilistic graphical models of the drone-payload system dynamics. At each time step $t$, the system state evolves as a function of the current state $\mathbf{s}_t$, action $\mathbf{a}_t$, function parameters $\theta$, and dynamics variable $\mathbf{z}_k$ which encodes the $k$'th payload's idiosyncrasies. Shaded nodes are observed. At training time, the factors of variation between payloads might be known (Fig. 4a) or unknown (Fig. 4b) while training model's parameters $\theta$. Regardless of the training regime, test-time $\mathbf{z}^{\text{test}}$ is always unknown (Fig. 4c), which we infer given the trained (fixed) model parameters $\theta^*$.

### 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 $\mathbf{z}^{\text{test}}$ in order to improve the learned dynamics model $p_{\theta^*}$ and the resulting MPC planner. Inference is performed by accumulating transitions $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ into $\mathcal{D}^{\text{test}}$, and using this data and the meta-trained model parameters $\theta^*$ to infer the current value of $\mathbf{z}^{\text{test}}$ in real time, as seen in the right side of Figure 3. A summary of the variables involved in the inference task is given by Figure 4c.

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

$$q_{\phi^{\text{test}}}(\mathbf{z}^{\text{test}}) \;=\; \mathcal{N}(\mu^{\text{test}}, \Sigma^{\text{test}}) \;\approx\; p(\mathbf{z}^{\text{test}}|\mathcal{D}^{\text{test}})\,, \quad (8)$$

parameterized by $\phi^{\text{test}} \doteq \{\mu^{\text{test}}, \Sigma^{\text{test}}\}$. Regardless of training regime (§IV-B or §IV-C), inferring $\mathbf{z}^{\text{test}}$ 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 $q_{\phi^{\text{test}}}(\mathbf{z}^{\text{test}})$ is close to the true distribution $p(\mathbf{z}^{\text{test}}|\mathcal{D}^{\text{test}})$, measured by the

Kullback-Leibler divergence:

$$\begin{aligned}\phi^* &\doteq \operatorname*{argmax}_{\phi} \; -\mathrm{KL}(q_\phi(\mathbf{z}^{\text{test}}) \| p(\mathbf{z}^{\text{test}} | \mathcal{D}^{\text{test}}, \theta^*)) \\ &= \operatorname*{argmax}_{\phi} \mathbb{E}_{\mathbf{z}^{\text{test}} \sim q_\phi} \log p(\mathbf{z}^{\text{test}} | \mathcal{D}^{\text{test}}, \theta^*) - \log q_\phi(\mathbf{z}^{\text{test}}) \\ &= \operatorname*{argmax}_{\phi} \mathbb{E}_{\mathbf{z}^{\text{test}} \sim q_\phi} \log p(\mathcal{D}^{\text{test}} | \mathbf{z}^{\text{test}}, \theta^*) - \log q_\phi(\mathbf{z}^{\text{test}}) + \log p(\mathbf{z}^{\text{test}}) \\ &= \operatorname*{argmax}_{\phi} \mathbb{E}_{\mathbf{z}^{\text{test}} \sim q_\phi} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}^{\text{test}}} \log p_{\theta^*}(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t, \mathbf{z}^{\text{test}}) - \mathrm{KL}(q_\phi(\mathbf{z}^{\text{test}}) \| p(\mathbf{z}^{\text{test}})) \\ &= \operatorname*{argmax}_{\phi} \; \mathrm{ELBO}(\mathcal{D}^{\text{test}} | \theta^*, \phi) \,. \end{aligned} \tag{9}$$

Note the objective (9) corresponds to the test-time ELBO of $\mathcal{D}^{\text{test}}$, analogous to training-time ELBO of $\mathcal{D}^{\text{train}}$ (6). Thus (9) scores how well $\phi$ describes the new data $\mathcal{D}^{\text{test}}$, 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 Summary

A summary of the full training and test-time procedure is provided in both Figure 3 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 $\{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\}$ are recorded into the corresponding task dataset, as well as the dynamics variable $\mathbf{z}_k$ if it is known (§IV-B). We then train the dynamics model $p_{\theta^*}$ using the dataset $\mathcal{D}^{\text{train}}$ via (7).

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

### F. Method Implementation

The quadcopter we use is the DJI Tello (Figure 1), which enables rapid experimentation for suspended payload control due to its small $98\,\mathrm{mm} \times 93\,\mathrm{mm} \times 41\,\mathrm{mm}$ size, light $80\,\mathrm{g}$ weight, long 13 minute battery life, and powerful motors. In our tasks, 3D printed payloads weighing between 10–15 grams are attached to the Tello via a string. Our experiments vary primarily the string length between 18–30cm long since the dynamics are more sensitive to string length than mass. We found that this range of string lengths exhibits a larger variation in dynamics while staying in the field of view of our external camera and not interfering with onboard altitude estimation.

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 [4]. 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 $\mathbf{s} \in \mathbb{R}^3$, which are saved every control step into the corresponding dataset in $\mathcal{D}^{\text{train}}$. In our comparative experiments, the final dataset $\mathcal{D}^{\text{train}}$ consisted of approximately 16,000 data points (1.1 hours of flight) split between 18cm and 30cm, which were then used by our meta-learning for model-based reinforcement learning algorithm.

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. MPC is run on an external laptop, with a time horizon of 5 steps, and we used the cross entropy method [3] to optimize, with a sample size 50, selecting 10 elite samples, and 3 iterations. This computation takes 50-100ms, so we selected our control frequency to be 4Hz for both training and test time, to allow the remaining 150-200ms for latent variable inference at each step. We adapted code from a PyTorch implementation of PETS [5] found https://github.com/quanvuong/handful-of-trials-pytorch.

**Algorithm 2** Model-Based Meta-Reinforcement Learning for Quadcopter Payload Transport

*// Training Phase*
**for** Task $k = 1$ to $K$ **do**
  **for** Time $t = 0$ to $T$ **do**
    Execute action $\mathbf{a}_t$ from human pilot
    **if** $\mathbf{z}_k$ is known **then** ▷ case §IV-B
      Record outcome: $\mathcal{D}^{\text{train}} \leftarrow \mathcal{D}^{\text{train}} \cup \{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{z}_k\}$
    **else** ▷ case §IV-C
      Record outcome: $\mathcal{D}^{\text{train}} \leftarrow \mathcal{D}^{\text{train}} \cup \{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\}$
Train dynamics model $p_{\theta^*}$ given $\mathcal{D}^{\text{train}}$ ▷ see (7)

*// Test Phase*
Initialize variational parameters: $\phi^* \leftarrow \{\mu^{\text{test}} = 0, \Sigma^{\text{test}} = I\}$
**for** Time $t = 0$ to $T$ **do**
  Solve optimal action $\mathbf{a}_t^*$ given $p_{\theta^*}$, $q_{\phi^*}$, and MPC ▷ see (2)
  Execute action $\mathbf{a}_t^*$
  Record outcome: $\mathcal{D}^{\text{test}} \leftarrow \mathcal{D}^{\text{test}} \cup \{\mathbf{s}_t, \mathbf{a}_t^*, \mathbf{s}_{t+1}\}$
  Infer variational parameters $\phi^*$ given $\mathcal{D}^{\text{test}}$ ▷ see (9)

## 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].

In these 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 conditioned on a history of states and actions?
- **Q3** How does our approach with known versus unknown dynamics variables compare?
- **Q4** Can we *generalize* to payloads that were not seen at training time?

[1]https://sites.google.com/view/ral-meta-rl-for-flight

TABLE I: Comparative evaluation of our method for the tasks of following a circle, square or figure-8 trajectory with either an 18cm or 30cm payload cable length. The table entries specify the average pixel tracking error over 5 trials, with $\infty$ denoting when all trials failed the task by deviating outside of the camera field of view. Note that the cable length was not given to any method *a priori*, and therefore online adaptation was required in order to successfully track the specified path. Our method was able to most closely track all specified paths for all payloads.

| Algorithm | Avg. Tracking Error (pixels) for each Task Path and Payload String Length (cm) | | | | | |
|---|---|---|---|---|---|---|
| | Circle | | Square | | Figure-8 | |
| | 18 | 30 | 18 | 30 | 18 | 30 |
| Ours (unknown variable) | **23.62±2.67** | **24.41±3.90** | **23.88±2.81** | **26.57±3.84** | **24.67±1.33** | 29.08±6.00 |
| Ours (known variable) | 31.81±6.49 | 30.49±2.65 | 26.37±3.63 | 31.68±4.68 | 29.84±2.84 | **28.28±3.76** |
| MBRL | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| MBRL with history | 39.96±4.40 | 42.36±2.84 | 32.37±2.40 | 39.26±5.16 | 34.17±1.90 | 41.01±7.26 |
| PID controller | 70.58±4.01 | 67.98±2.50 | 65.79±9.99 | 69.53±6.85 | 90.15±10.40 | 86.37±9.27 |

TABLE II: Generalization results. Training data consists of 18cm & 30cm cable lengths augmented with only a few minutes of 24cm data. At test-time, we use 21cm and 27cm cables, to test the model's ability to adapt to new dynamics at intermediate cable lengths.

| Algorithm | Avg. Tracking Error (pixels) for each Task Path and Payload String Length (cm) | | | | | |
|---|---|---|---|---|---|---|
| | Circle | | Square | | Figure-8 | |
| | 21cm | 27cm | 21cm | 27cm | 21cm | 27cm |
| Ours (3 unknown variables) | **16.8±1.3** | **21.7±2.1** | **24.1±2.4** | **24.3±1.4** | **36.2±2.5** | **40.1±3.8** |
| MBRL with history | 21.1±4.7 | 25.9±1.6 | 28.3±2.5 | 37.0±2.6 | 43.8±3.7 | 41.9±3.9 |

**Q5** Is the test-time inference procedure able to differentiate between different *a priori* unknown payloads?

**Q6** Can our approach enable a quadcopter to fulfill a complete payload pick-up, transport, and drop-off task, as well as other realistic payload transportation scenarios?

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*, in which the state consists of only the current payload pixel location and size.
- *MBRL with history*, 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.

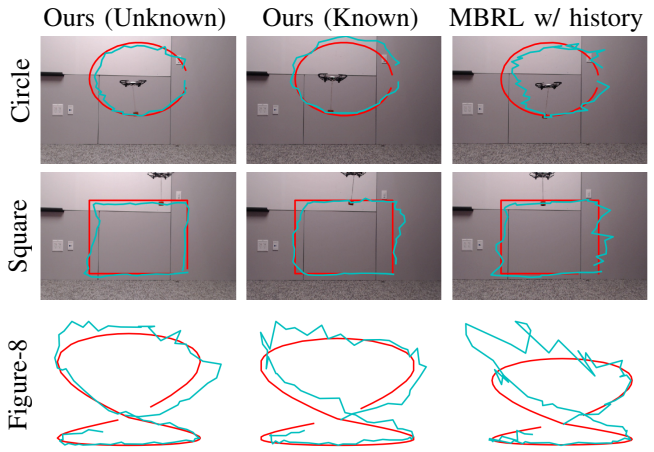



Fig. 5: Comparison of our meta-learning approach with unknown and known factors of variation versus model-based reinforcement learning (MBRL) conditioned on a history of states and actions. The tasks are to either follow a circle or square in the image plane, or a figure-8 parallel to the ground. The 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.

### A. 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 (Figure 5) in the image plane or a figure-8 path parallel to the ground, and with a suspended cable either 18cm or 30cm long. Given this single factor of variation, we used a latent variable of dimension one. Although the training data included payloads with these cable lengths, the length was unknown to all methods during test-time experiments.

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 Figure 5. Both the online adaptation methods—our approach and MBRL with history—better track the specified goal trajectories compared to the non-adaptation methods—MBRL 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 with history (**Q2**). Interestingly, our approach with unknown latent variables at training time outperformed our approach with known latent variables (**Q3**). A possible explanation is that inferring unknown latent variables at training-time captures unspecified types of variation from potentially hard to observe factors, in comparison to specifying observable types of variation (e.g. tether length). In addition, sampling latent variables from a distribution with full support during training could prevent test set latent samples from being out of distribution. Nevertheless, this shows our approach does not require *a priori* knowledge of latent factors during training to successfully adapt at test time. We also demonstrate our method's ability to generalize to new payloads not seen during training (**Q4**). While we found training using only two string lengths was sufficient to learn to adapt to either dynamics, the ability to generalize improves with several example tasks. In our case, learning how string lengths affect the dynamics benefited from a few minutes of data from a third (24cm) string length, allowing us to rapidly interpolate to unseen string

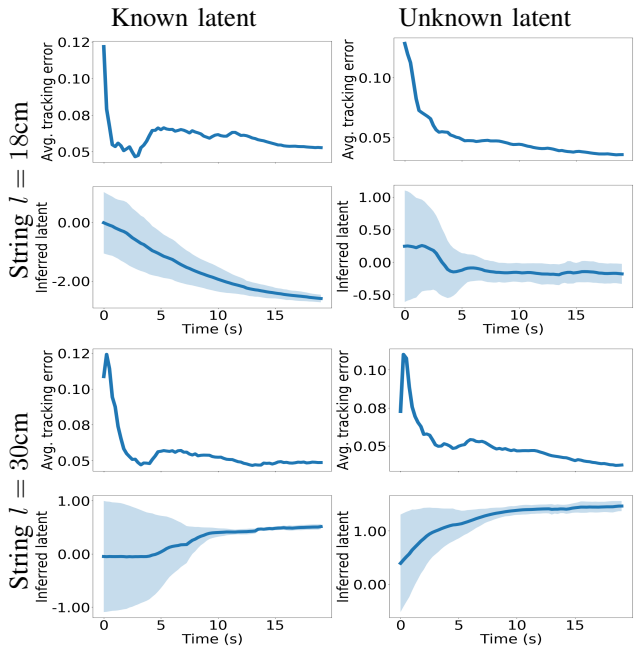



Fig. 6: Visualization of the inferred latent variable and tracking error over time for the task of following a figure-8 trajectory. We show our approach trained with known variables (left column) and unknown variables (right column) with either a payload cable length of 18 cm (top row) or 30 cm (bottom row). For all approaches, the inferred latent variable converges as the quadcopter flies and adapts online. The converged final latent values are different depending on the cable length, which shows the online adaptation mechanism is able to automatically differentiate between the different payloads. Furthermore, as the latent value converges, the tracking error also reduces, which demonstrates that there is a correlation between inferring the correct latent variable and the achieved task performance.

lengths of 21cm or 27cm at test-time, shown in Table II.

Figure 6 and Figure 8 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 (**Q5**). 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**).

### B. End-to-End Payload Transportation

We also evaluated our approach on an end-to-end payload transportation task (Figure 1), 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. Figure 7 shows our approach successfully completes the full task (**Q6**) due to our online adaptation mechanism (**Q1**), which enables the drone to follow trajectories better and pick up the payload by automatically inferring whether the payload is attached or detached (**Q5**). Furthermore, the continuous nature of this task 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.

### C. Additional Use Cases

In addition to enabling trajectory following and end-to-end payload transportation, we demonstrated our approach to transport a suspended payload (**Q6**): towards a moving target, around an obstacle by following a predefined path, and along trajectories dictated using a "wand"-like interface (see videos and website).

## 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 with minimal state information. 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.

## 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. *Conference on Decision and Control*, pages 4653–4660, 2016.

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

[3] Zdravko I Botev, Dirk P Kroese, Reuven Y Rubinstein, and Pierre L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pages 35–59. Elsevier, 2013.

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

[5] 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*, pages 4754–4765, 2018.

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

[7] 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.

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

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

[10] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

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

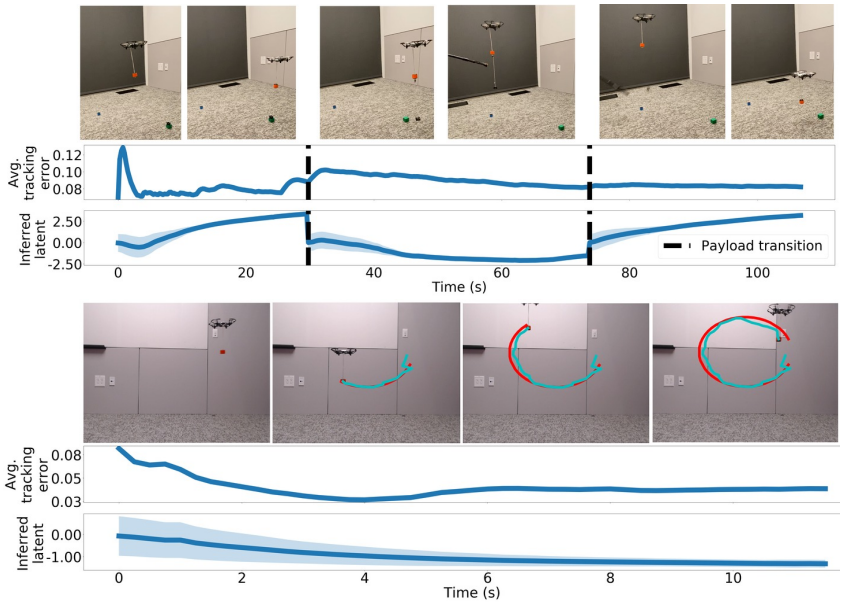



Fig. 7: Visualization of our approach successfully completing the full quadcopter payload transportation task. The task consists of three phases: the quadcopter before picking up the payload; while the payload is in transit to the goal; and after the payload is dropped off. Our approach continuously adapts the latent dynamics variable online using the current test-time dataset, flushing the test-time dataset each time the quadcopter transitions between phases, shown by vertical black lines. Note 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.




Fig. 8: 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.


[12] Petros Ioannou and Bariş Fidan. *Adaptive control tutorial*. SIAM, 2006.

[13] 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.

[14] 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*, 2018.

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

[16] Rituraj Kaushik, Timothée Anne, and Jean-Baptiste Mouret. Fast online adaptation in robotics through meta-learning embeddings of simulated priors. In *Intelligent Robots and Systems (IROS)*, pages 5269–5276, 2020.

[17] Andras Gabor Kupcsik, Marc Peter Deisenroth, Jan Peters, Gerhard Neumann, et al. Data-efficient generalization of robot skills with contextual policy search. In *Association for the Advancement of Artificial Intelligence*, pages 1401–1407, 2013.

[18] 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. *Robotics and Automation Letters*, 4(4):4224–4230, 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2930489.

[19] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *International Conference on Robotics and Automation*, pages 1642–1648, 2010.

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

[21] 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.

[22] 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.

[23] Anusha Nagabandi, Gregory Kahn, Ronald 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*, pages 7559–7566, 2018.

[24] Anusha Nagabandi, Ignasi Clavera, Simin Liu, and Ronald Fearing. Learning to adapt in dynamic, real-world environments via meta-reinforcement learning. In *International Conf. on Learning Representations*, 2019.

[25] 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.

[26] 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.

[27] 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.

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

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

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

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

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

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

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

[35] 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.