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

Anonymous Author(s)
Affiliation
Address
`email`

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 adapt models of altered dynamics within seconds after picking up or dropping a payload. Our experiments demonstrate that our approach outperforms non-adaptive methods on several challenging suspended payload transportation tasks. Videos available at: https://sites.google.com/view/corl-meta-rl-for-flightsites.google.com/view/corl-meta-rl-for-flight

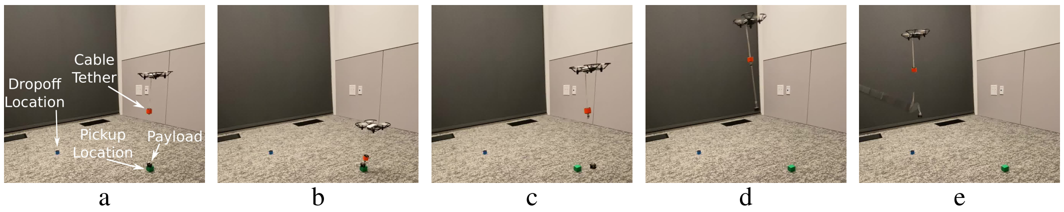


Figure 1: Our meta-reinforcement learning method controlling a quadcopter transporting a suspended payload. This task is challenging since each payload induces different system dynamics, which requires the quadcopter controller to adapt online. The controller learned via our meta-learning approach can (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.

1 Introduction

System identification is a powerful tool to control well-characterized robotic systems, such as quadcopters, in the absence of any physical interaction with the environment. 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. Unfortunately, it is precisely physical interactions with movable objects 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.

In this work, we specifically investigate how to enable a quadcopter to control a suspended payload, in which the 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 system for quadcopter control with suspended payloads. 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.

We demonstrate in our experiments that our method enables a quadcopter to plan and execute trajectories that follow desired payload trajectories, drop off these payloads at designated 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.

2 Related Work

Prior work on control for aerial vehicles has demonstrated impressive performance and agility in navigating between small openings [19], aerobatics [17], and obstacle avoidance [25]. These approaches have also enabled aerial vehicles to aggressively control suspended payloads [29, 30]. These methods typically rely on manual system identification, where an expert derives equations of motion and any physical parameters are estimated for both the aerial vehicle [18, 33] and the suspended payload [29, 30]. Although these approaches have successfully enabled controlled flight of the hybrid system, they require *a priori* knowledge of the system [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 adapted online according to a specified error metric [28, 12]. 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. 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 [31] or policy gradient methods that directly learn a control policy [32]. Although these model-free approaches have been used to learn policies for various tasks [20, 27], 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 [5, 9, 21, 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 [1, 16], 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 [8, 11, 22] or input variables [24, 26]. There has been prior work on model-based meta-learning for quadcopters. O'Connell et al. [23] 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. [22] 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.

3 Preliminaries

We first introduce our notation, problem formulation, and preliminaries on model-based reinforcement learning (MBRL) that our meta-learning algorithm builds upon. Conventionally, a control problem can be represented as a Markov decision process, with state $\mathbf{s} \in \mathbb{R}^{d_\mathbf{s}}$, 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 learns the underlying dynamics from data, with minimal prior knowledge of the system. We can train a dynamics model $p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ with parameters θ by collecting data in the real world, which we can view as sampling "ground truth" tuples $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. Given a dataset $\mathcal{D}^{\text{train}} = \{(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1), (\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2), ...\}$, the parameters θ^* can be learned with maximum likelihood:

$$\theta^* = \arg\max_\theta \; p(\mathcal{D}^{\text{train}}|\theta) = \arg\max_\theta \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}^{\text{train}}} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t). \quad (1)$$

The specific model-based RL method we build on is the PETS algorithm [4], which has previously been shown to handle expressive neural network dynamics models and attain good sample efficiency and 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, 21]. MPC uses the dynamics model to predict into the future, and selects the action sequence that has the highest predicted reward:

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

where $\mathbf{s}_\tau$ is recursively sampled from the 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 method is provided in Algorithm 1, and we refer the reader to Chua et al. [4] for additional details.

Algorithm 1 Model-Based RL (PETS)

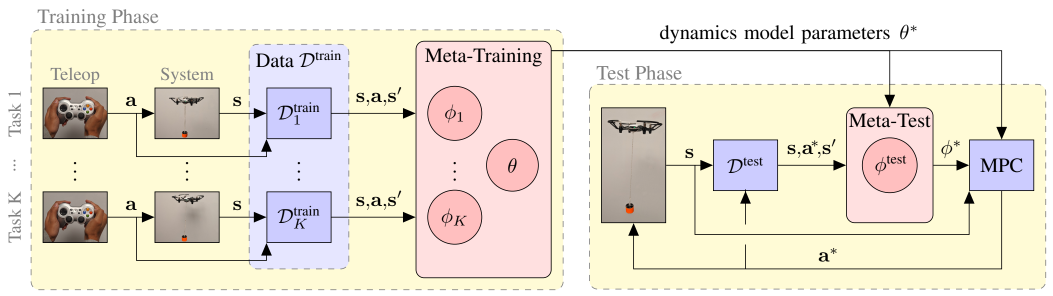
```
1: Initialize dynamics model $p_\theta$ with random parameters $\theta$
2: while not done do
3:     Get current state $\mathbf{s}_t$
4:     Decide action $\mathbf{a}_t^*$ given $p_{\theta^*}$ and $\mathbf{s}_t$    ▷ see (2)
5:     Execute action $\mathbf{a}_t^*$
6:     Record: $\mathcal{D}^{\text{train}} \leftarrow \mathcal{D}^{\text{train}} \cup \{\mathbf{s}_t, \mathbf{a}_t^*, \mathbf{s}_{t+1}\}$
7:     Train dynamics model $p_\theta$ with $\mathcal{D}^{\text{train}}$ ▷ see (1)
8: end while
```

4 Model-Based Meta-Learning for Quadcopter Payload Transport

Our goal is to enable a quadcopter to pick up and transport payloads with *a priori* unknown properties. We formulate this problem as a *visual servoing* task: a fixed camera localizes the payload in image space, and the user specifies a desired flight trajectory, also in image space. The state $\mathbf{s}$ corresponds only to the location and size of the payload in the image.

The primary challenge is that this interaction is difficult to model *a priori* because each suspended payload has 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 $\mathbf{s}$, such as the unknown properties of the payload. We therefore approach this problem 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 trajectory (Fig. 1). First, the quadcopter must fly to the location of the payload (Fig. 1a), attach itself to the payload using a suspended cable (Fig. 1b), and then lift the payload off the ground (Fig. 1c).



Training Phase
Teleop
System
Data $\mathcal{D}^{\text{train}}$
Meta-Training
Task 1
Task K
a
s
$\mathcal{D}_1^{\text{train}}$
$\mathcal{D}_K^{\text{train}}$
s,a,s′
ϕ_1
ϕ_K
θ
dynamics model parameters θ^*
Test Phase
$\mathcal{D}^{\text{test}}$
s,a*,s′
Meta-Test
ϕ^{test}
ϕ^*
MPC
a*

Figure 2: 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 θ and the adaptation parameters $\phi_{1:K}$ for each payload task. At test time, using the learned dynamics model parameters θ^*, the robot infers the optimal latent variable ϕ^* online using all of the data $\mathcal{D}^{\text{test}}$ from the current task. The dynamics model, parameterized by θ^* and ϕ^*, 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.

Finally, the quadcopter attempts to reach the goal destination and releases the payload (Fig. 1e). The quadcopter is then able to continue transporting other payloads in a similar manner. Each payload transition can alter the dynamics drastically.

4.1 Data Collection

We first collect data by manually piloting the quadcopter (Fig. 2, left) along random paths for each of the K different suspended payloads. 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. The quadcopter we use is the DJI Tello (Fig. 1). The Tello is ideal for 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.

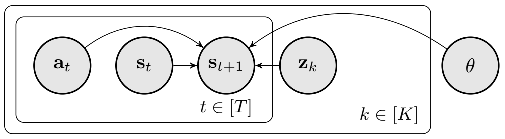
During data collection, we record the controls (Cartesian velocity commands $\mathbf{a} \in \mathbb{R}^3$) and the payload location (pixel location and size of the payload in image space, $\mathbf{s} \in \mathbb{R}^3$), which we track with an externally mounted RGB camera using OpenCV [3]. Observed states are stored every 0.25 seconds into $\mathcal{D}^{\text{train}}$. To model how past states and actions affect the future trajectory of the payload, we modify the state to include the past 8 states and actions, resulting in the state having dimension $\mathbf{s} \in \mathbb{R}^{48}$. The final dataset $\mathcal{D}^{\text{train}}$ 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.

4.2 Model Training with Known Dynamics Variables

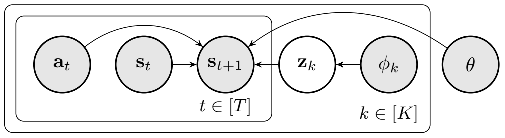
As a warmup, we first discuss the case where we know all the "factors of variation" in the dynamics across tasks, represented explicitly as a "dynamics variable" $\mathbf{z}_k \in \mathbb{R}^{d_\mathbf{z}}$ that is known at training time, but unknown at test-time. For example, we might know that the source of variation is the tether length L, and therefore we would specify $\mathbf{z}_k \leftarrow L_k \,\forall\, k$ at training time. In §4.3, we will address the case where these factors of variation are completely unknown. With known $\mathbf{z}_k$, we can learn a single dynamics model p_θ across all tasks by using $\mathbf{z}_k$ as an auxiliary input: $\mathbf{s}_{t+1} \sim p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_k)$. 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 $\mathbf{s}$. Training is therefore analogous to (1), but with an additional conditioning on $\mathbf{z}_{1:K} \doteq [\mathbf{z}_1, ..., \mathbf{z}_K]$:

$$\theta^* \doteq \arg\max_\theta \log p(\mathcal{D}^{\text{train}}|\mathbf{z}_{1:K}, \theta) = \arg\max_\theta \sum_{k=1}^{K} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}^{\text{train}}_k} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_k)\,. \quad (3)$$

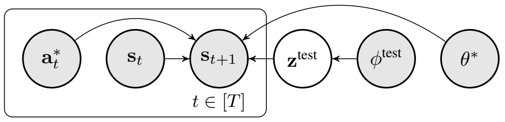
The variables in this training process are summarized in the graphical model shown in Fig. 3a, in which every variable is observed except for the "true" model parameters θ, which we infer approximately as θ^* using maximum likelihood estimation in (3).



(a) **Training**-time with **known** payload properties $\mathbf{z}_k$.



(b) **Training**-time with **unknown** payload prop. $\mathbf{z}_k$.



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

Figure 3: Graphical models of the drone-payload system dynamics. At each time step t, the system evolves as a function—parameterized by θ—of the current state $\mathbf{s}_t$, action $\mathbf{a}_t$, and the k'th payload's idiosyncrasies (modelled as $\mathbf{z}_k$). Shaded nodes are observed. At training time, the payload might be known (Fig. 3a) or unknown (Fig. 3b). Either way, test-time $\mathbf{z}^{\text{test}}$ is always unknown (Fig. 3c), and inferred given the trained dynamics model parameters θ^*.

Algorithm 2 Model-Based Meta RL for Quadcopter Payload Transport

```
 1: // Training Phase
 2: for Task $k = 1$ to $K$ do
 3:     for Time $t = 0$ to $T$ do
 4:         Get action $\mathbf{a}_t$ from human pilot
 5:         Execute action $\mathbf{a}_t$
 6:         if $\mathbf{z}_k$ is known then                         ▷ case §4.2
 7:             Record: $\mathcal{D}^{\text{train}} \leftarrow \mathcal{D}^{\text{train}} \cup \{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{z}_k\}$
 8:         else                                                 ▷ case §4.3
 9:             Record: $\mathcal{D}^{\text{train}} \leftarrow \mathcal{D}^{\text{train}} \cup \{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\}$
10:         end if
11:     end for
12: end for
13: Train dynamics model $p_{\theta^*}$ with $\mathcal{D}^{\text{train}}$     ▷ see (5)
14:
15: // Test Phase
16: Initialize $\phi^* \leftarrow \{\mu^{\text{test}} = 0, \Sigma^{\text{test}} = I\}$
17: for Time $t = 0$ to $T$ do
18:     Decide action $\mathbf{a}_t^*$ given $p_{\theta^*}$, $q_{\phi^*}$          ▷ see (2)
19:     Execute action $\mathbf{a}_t^*$
20:     Record: $\mathcal{D}^{\text{test}} \leftarrow \mathcal{D}^{\text{test}} \cup \{\mathbf{s}_t, \mathbf{a}_t^*, \mathbf{s}_{t+1}\}$
21:     Infer payload params $\phi^*$ given $\mathcal{D}^{\text{test}}$ ▷ see (6)
22: end for
```

4.3 Meta-Training with Latent Dynamics Variables

The formulation in §4.2 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 θ *jointly*, as shown by Fig. 3b, without needing 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 $p(\theta, \mathbf{z}_{1:K}|\mathcal{D}^{\text{train}}_{1:K})$. 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 $p(\theta, \mathbf{z}_{1:K}|\mathcal{D}^{\text{train}}_{1:K})$ 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, $q_{\phi_k}(\mathbf{z}_k) = \mathcal{N}(\mu_k, \Sigma_k) \approx p(\mathbf{z}_k|\mathcal{D}^{\text{train}}) \, \forall \, k \in [K]$, 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 $\mathcal{D}^{\text{train}} = \mathcal{D}^{\text{train}}_{1:K}$, analogous to Equation (3). The only difference to §4.2 is that we must (approximately) marginalize out $\mathbf{z}_{1:K}$ because it is unknown:

$$\log p(\mathcal{D}^{\text{train}}|\theta) = \log \int_{\mathbf{z}_{1:K}} p(\mathcal{D}^{\text{train}}|\mathbf{z}_{1:K}, \theta) p(\mathbf{z}_{1:K}) \mathrm{d}\mathbf{z}_{1:K} = \sum_{k=1}^{K} \log \mathbb{E}_{\mathbf{z}_k \sim q_{\phi_k}} p(\mathcal{D}^{\text{train}}|\mathbf{z}_k, \theta) \cdot \frac{p(\mathbf{z}_k)}{q_{\phi_k}(\mathbf{z}_k)}$$

$$\geq \sum_{k=1}^{K} \mathbb{E}_{\mathbf{z}_k \sim q_{\phi_k}} \left[\sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}^{\text{train}}_k} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_k) \right] - \mathrm{KL}(q_{\phi_k}(\mathbf{z}_k)||p(\mathbf{z}_k)) \doteq \mathrm{ELBO}(\mathcal{D}^{\text{train}}|\theta, \phi_{1:K}). \tag{4}$$

This is the evidence lower bound (ELBO), which 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 θ and the variational parameters $\phi_{1:K}$ of each task with respect to the evidence lower bound:

$$\theta^* \doteq \arg\max_{\theta} \max_{\phi_{1:K}} \text{ELBO}(\mathcal{D}^{\text{train}}|\theta, \phi_{1:K}). \tag{5}$$

4.4 Test-Time Task Inference

At test time, the robot must adapt online to the new task—such as a new payload—by inferring the unknown dynamics variables $\mathbf{z}^{\text{test}}$ in order to improve the learned dynamics model p_{θ^*} 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 θ^* to infer the current value of $\mathbf{z}^{\text{test}}$ in real time, as seen in the right side of Fig. 2. A summary of the variables in the inference task is given by Fig. 3c. Similarly to §4.3, exact inference is intractable, and we therefore use a variational approximation for $\mathbf{z}^{\text{test}}$: $q_{\phi^{\text{test}}}(\mathbf{z}^{\text{test}}) = \mathcal{N}(\mu^{\text{test}}, \Sigma^{\text{test}}) \approx p(\mathbf{z}^{\text{test}}|\mathcal{D}^{\text{test}})$, parameterized by $\phi^{\text{test}} \doteq \{\mu^{\text{test}}, \Sigma^{\text{test}}\}$. Regardless of training regime (§4.2 or §4.3), 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 ϕ^{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:

$$\phi^* \doteq \text{argmax}_{\phi} \; - \text{KL}(q_{\phi}(\mathbf{z}^{\text{test}})||p(\mathbf{z}^{\text{test}}|\mathcal{D}^{\text{test}}, \theta^*)) = \arg\max_{\phi} \text{ELBO}(\mathcal{D}^{\text{test}}|\theta^*, \phi), \tag{6}$$

with proof in appendix, equation (7). Note the objective (6) corresponds to the test-time ELBO of $\mathcal{D}^{\text{test}}$, analogous to training-time ELBO of $\mathcal{D}^{\text{train}}$ (4). Thus (6) scores how well ϕ describes the new data $\mathcal{D}^{\text{test}}$, under our variational constraint that q is assumed to be Gaussian. Since θ^* was already inferred at training time, we treat it as a constant during this test-time optimization. Equation (6) is tractable to optimize, and therefore at test time we perform gradient descent online in order to learn ϕ^{test} and therefore improve the predictions of our learned dynamics model.

4.5 Method Summary

A summary of the full training procedure and test-time visual servoing evaluation is provided in both Fig. 2 and Algorithm 2. During the training phase, dataset $\mathcal{D}^{\text{train}}$ is gathered for K tasks consisting of tuples $\{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\}$, as well as the dynamics variable $\mathbf{z}_k$ if it is known (§4.2). We then train the dynamics model p_{θ^*} using the dataset $\mathcal{D}^{\text{train}}$ via (5). 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 dynamics $\mathbf{z}^{\text{test}}$. At each time step, we solve for the optimal action $\mathbf{a}_t^*$ given p_{θ^*} and the current estimate of $\mathbf{z}^{\text{test}}$ using MPC. The quadcopter executes this action and records the transition $\{\mathbf{s}_t, \mathbf{a}_t^*, \mathbf{s}_{t+1}\}$ in the test dataset $\mathcal{D}^{\text{test}}$. We then adapt the latent variable 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.

5 Experimental Evaluation

We now present an experimental evaluation of our meta-learning approach in the context of quadcopter suspended payload control tasks. The experiments are best seen in the accompanying video [1], which also include additional evaluations.

We evaluated our meta-learning approach with both known variables (§4.2) and latent variables (§4.3), and compared to the three following approaches. **MBRL without history** has state consisting of only the current payload pixel location and size. **MBRL with history**, a baseline meta-learning approach that concatenates past eight states and actions together as input to the dynamics model. While concatenating past states often helps with non-Markovian dynamics [16], prior methods also use histories of recent states to understand changes in the dynamics function, using either an RNN [6, 22] or a MAML-based method [22]. Lastly, **PID controller** uses three PID controllers, one for each Cartesian velocity command axis. We manually tuned the PID gains based on the performance of the controller on a trajectory not used in our experiments for a single payload.

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? **Q3**: Can we *generalize* to payloads that were not seen at training time? **Q4**: Is our test-time inference procedure able to differentiate between different *a*

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

priori unknown payloads? **Q5**: 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?

Trajectory Following We first evaluate the ability of our method to track specified payload trajectories in isolation, separately from the full 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.

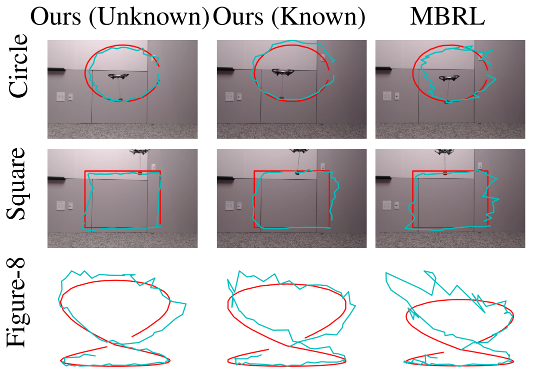


Figure 4: Comparison of our meta-learning approach with unknown and known factors of variation versus model-based reinforcement learning (MBRL) with past states and actions concatenated. The tasks are to either follow a circle or square in the image plane, or a figure-8 parallel to the ground. The specified 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.

Table 1 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 MBRL (**Q2**). Our approach meta-trained with unknown latent dynamics variables also performs similarly to our approach trained with known dynamics models, which highlights that our approach does not require *a priori* knowledge of the payloads during training to successfully adapt at test time. It can also generalize to latent variables not seen during training (**Q3**), rapidly adapting to string lengths of 21cm or 27cm, shown in Table 2.

Fig. 5 and Fig. 7 (appendix) show that the dynamics variable converges to different values depending on the cable length, which shows that the test-time inference procedure can differentiate between different payload dynamics (**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**).

Table 1: Comparative evaluation for the tasks of following a circle, square or figure-8 trajectory with either an 18cm or 30cm payload cable length. Table entries specify the average pixel tracking error over 5 trials, with ∞ denoting when all trials failed the task by deviating outside of the camera field of view. Note 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	
	18cm	30cm	18cm	30cm	18cm	30cm
Ours (unknown variable)	**23.6±2.7**	**24.4±3.9**	**23.9±2.8**	**26.6±3.8**	**24.7±1.3**	29.1±6.0
Ours (known variable)	31.8±6.5	30.5±2.7	26.3±3.6	31.7±4.7	29.8±2.8	**28.3±3.8**
MBRL without history	∞	∞	∞	∞	∞	∞
MBRL	40.0±4.4	42.4±2.8	32.4±2.4	39.3±5.2	34.2±1.9	41.0±7.3
PID controller	70.6±4.0	68.0±2.5	65.8±10.0	69.5±6.9	90.2±10.4	86.4±9.3

Table 2: Generalization results. Training data comprises only 18cm, 24cm, and 30cm cable lengths. At test-time, we use 21cm and 27cm cables, to test the model's ability to interpolate between familiar 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	21.1±4.7	25.9±1.6	28.3±2.5	37.0±2.6	43.8±3.7	41.9±3.9

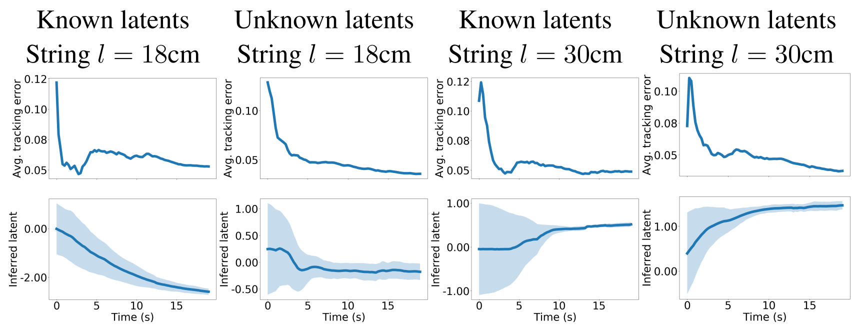


Figure 5: Visualization of the inferred latent variable and tracking error over time for the task of following a figure-8 trajectory. For all approaches, the inferred latent variable converges as the quadcopter flies and adapts online. The converged final latent values are different depending on the 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.

End-to-End Payload Transportation We also evaluated our approach on a full end-to-end payload transportation task (Fig. 6), 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. Our approach successfully completes the full task (**Q5**) due to our online adaptation mechanism (**Q1**), enabling 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. Additional tasks, including navigating around an obstacle (Fig. 8), greedily following a target (Fig. 9), and following trajectories dictated using a “wand”-like interface (Fig. 10), are available in the appendix.

6 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 to effectively visual servo. We augment this dynamics model with latent variables, which represent unknown factors of variation in the payload. These latent variables are inferred online to improve predictive accuracy 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.

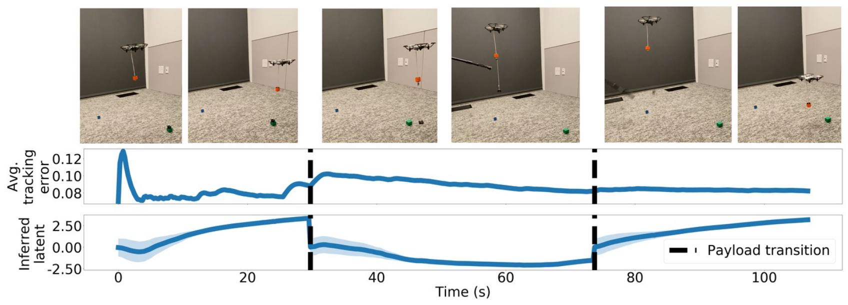


Figure 6: Our approach successfully completing the full quadcopter payload transportation task consisting of three phases: before the quadcopter picks up the payload, during quadcopter transit of the payload to the goal, and after the payload is dropped off. Our approach continuously infers the latent dynamics variable online using the current test-time dataset, and flushes the test-time dataset each time the quadcopter transitions between phases (shown by vertical black lines). The inferred latent variable is the same for when no payload is attached, but changes when the payload is attached, demonstrating 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.

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] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[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 (ICML)*, 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 Workshop on Meta-Learning*, 2018.

[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 (CoRL)*, 2018.

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

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

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

[33] 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.

A Variational Inference

At test time, we can approximately infer the relevant effects that our test-time payload is having on our system, using variational inference to optimize ϕ^{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:

$$\begin{aligned}
\phi^* &\doteq \arg\min_{\phi} \; \text{KL}(q_\phi(\mathbf{z}^{\text{test}})||p(\mathbf{z}^{\text{test}}|\mathcal{D}^{\text{test}}, \theta^*)) \\
&= \arg\max_{\phi} \; \mathbb{E}_{\mathbf{z}^{\text{test}}\sim q_\phi} \log p(\mathbf{z}^{\text{test}}|\mathcal{D}^{\text{test}}, \theta^*) - \log q_\phi(\mathbf{z}^{\text{test}}) \\
&= \arg\max_{\phi} \; \mathbb{E}_{\mathbf{z}^{\text{test}}\sim q_\phi} \log p(\mathcal{D}^{\text{test}}|\mathbf{z}^{\text{test}}, \theta^*) - \log q_\phi(\mathbf{z}^{\text{test}}) + \log p(\mathbf{z}^{\text{test}}) - \log p(\mathcal{D}^{\text{test}}|\theta^*) \\
&= \arg\max_{\phi} \; \mathbb{E}_{\mathbf{z}^{\text{test}}\sim q_\phi} \left[\sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})\in\mathcal{D}^{\text{test}}} \log p_{\theta^*}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}^{\text{test}}) \right] - \text{KL}(q_\phi(\mathbf{z}^{\text{test}})||p(\mathbf{z}^{\text{test}})), \\
&= \arg\max_{\phi} \; \text{ELBO}(\mathcal{D}^{\text{test}}|\theta^*, \phi).
\end{aligned} \tag{7}$$

As seen in (7), minimizing the Kullback-Leibler divergence is equivalent to maximizing the evidence lower bound (ELBO) of the data observed at test-time.

B 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-pytorchgithub.com/quanvuong/handful-of-trials-pytorch.

C Closed-loop performance

The dynamics variables are also interpretable: Fig. 7 shows the inferred dynamics variable and tracking error while our policy executes during test time. Note that the dynamics variable converges to different values depending on the cable length, which shows that the test-time inference procedure can differentiate between the dynamics of different payloads.

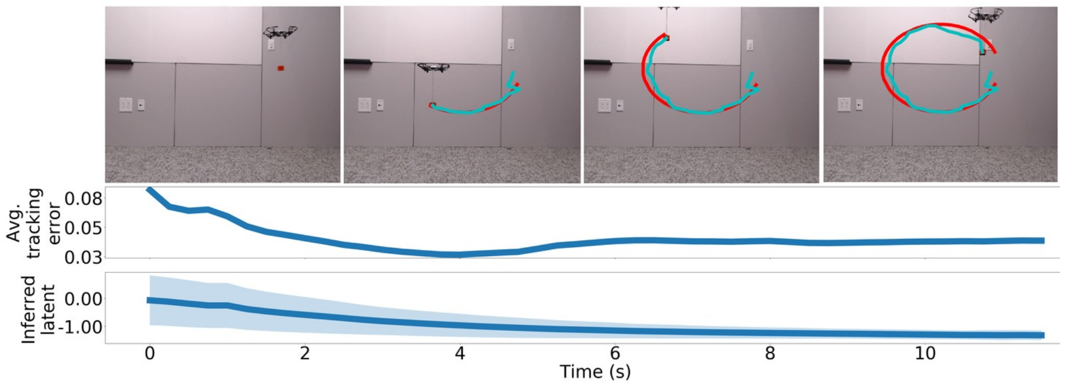


Figure 7: As the quadcopter follows the circle trajectory using our model-based controller, our approach adapts online to the *a priori* unknown payload by inferring the latent value which maximizes the dynamics models accuracy. This online adaptation reduces the tracking error as the quadcopter flies, enabling the quadcopter to successfully complete the task.

D Additional use cases

Our method also enables the quadcopter to perform other tasks, including navigating around an obstacle by following a predefined path (Fig. 8), greedily following a target (Fig. 9), and following along trajectories dictated using a "wand"-like interface (Fig. 10).

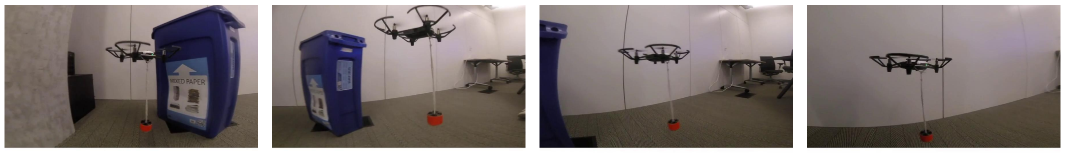


Figure 8: Our approach enables a quadcopter to transport a suspended payload around an obstacle. The user first defines a path that goes around the obstacle in the pixel space of the external camera. Our approach then encourages the suspended payload to follow this path while simultaneously adapting to the properties of the suspended payload.

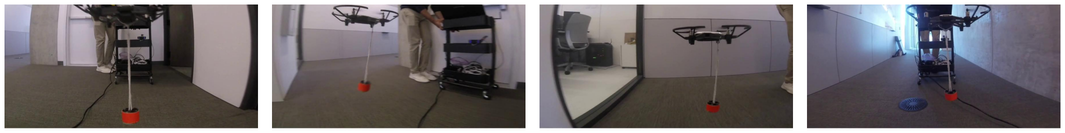


Figure 9: Our approach enables a quadcopter to control a suspended payload to follow a target. The target is the external camera that is used to track the suspended payload. Our approach encourages the suspended payload to stay in the center of the camera image and at a specific pixel size, and therefore as the external camera moves, the quadcopter moves in order to keep the suspended payload centered.

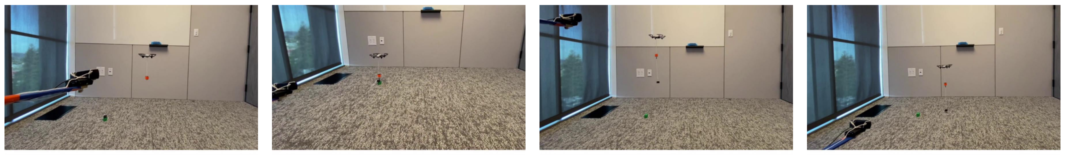


Figure 10: Our approach enables a quadcopter to follow trajectories dictated using a "wand"-like interface. The wand consists of mounting the external camera that is used to track the suspended payload on the end of a stick. By defining the cost function to encourage the suspended payload to stay centered, as the user moves the wand, our approach enables the quadcopter to adapt online to the specific payload while keeping the payload centered in the external camera's field-of-view.