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:

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

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 github.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.

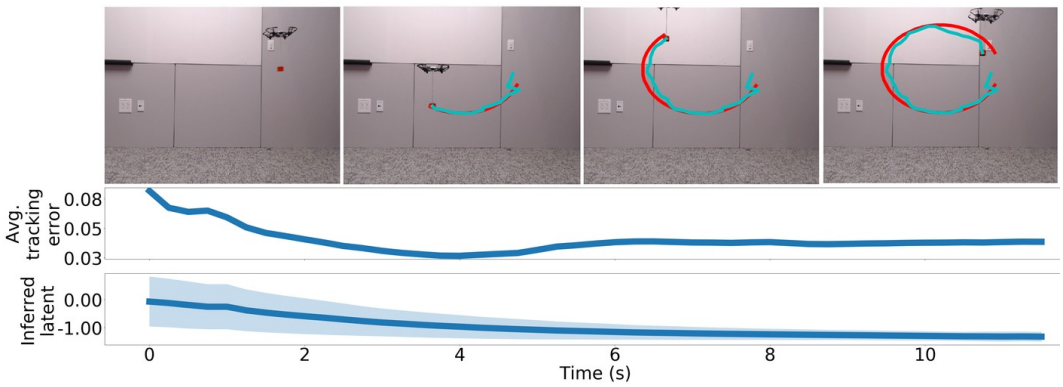

Figure 7: As the quadcopter follows the circle trajectory using our model-based controller, our approach adapts online to the *a priori* unknown payload by inferring the latent value which maximizes the dynamics models accuracy. This online adaptation reduces the tracking error as the quadcopter flies,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

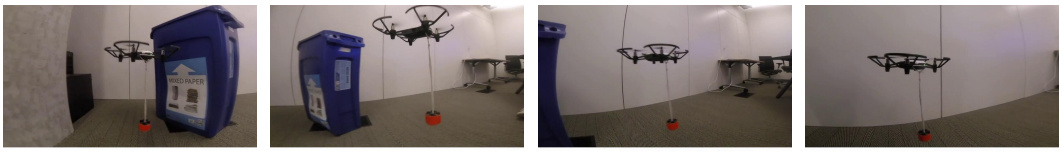

Figure 8: Our approach enables a quadcopter to transport a suspended payload around an obstacle. The user first defines a path that goes around the obstacle in the pixel space of the external camera. Our approach then encourages the suspended payload to follow this path while simultaneously adapting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

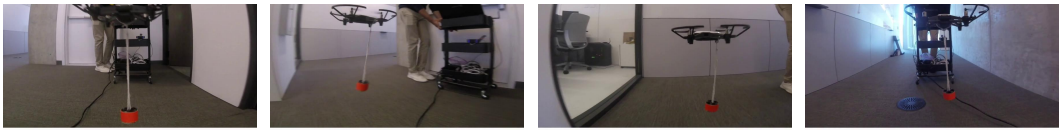

Figure 9: Our approach enables a quadcopter to control a suspended payload to follow a target. The target is the external camera that is used to track the suspended payload. Our approach encourages the suspended payload to stay in the center of the camera image and at a specific pixel size, and therefore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

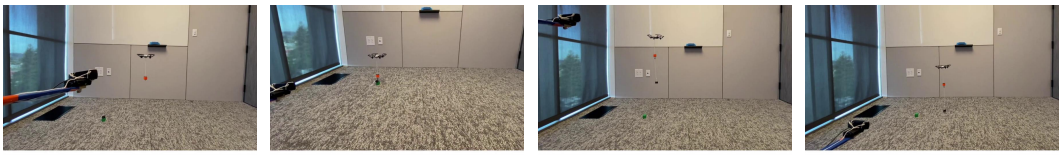

Figure 10: Our approach enables a quadcopter to follow trajectories dictated using a "wand"-like interface. The wand consists of mounting the external camera that is used to track the suspended payload on the end of a stick. By defining the cost function to encourage the suspended payload to stay centered, 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.