Low Level Control of a Quadrotor with Deep Model-Based Reinforcement Learning

Nathan O. Lambert and Isabella Huang

Extended Abstract

Physical robotic systems, especially at scales where significant dynamical uncertainties dominate, can be extremely difficult to model and control. Reinforcement learning techniques have proven to be successful in synthesizing controllers without prior system knowledge, but applying these techniques to real robots using low level sensing and actuation is underexplored. To this end, our work develops methods and techniques for the low level hover control of a mini Crazyflie quadrotor.

Model-based reinforcement learning (MBRL) is a data-efficient technique that draws from both on and off-policy experimental data to synthesize a forwards neural network dynamics model. In our setting, we bypassed the built-in Crazyflie PID controller and instead utilized only low level onboard sensors as input and motor pulse width modulation (PWM) signals as output. Architectural details of this dynamics model such as hyperparameters and state size were selected to minimize experimental prediction error. We demonstrated that we could use a simple discrete-action random shooting model predictive controller (MPC) based on its current dynamical model to iteratively gather experimental data and subsequently improve the fidelity of the dynamics model after each rollout. At the same time, the improvement of the dynamics model induced better MPC hovering control. To maximize the accuracy of our dynamics model, we leveraged a probabilistic loss function that penalized estimation variance to promote output stability, stacked prior observations to capture more context, pruned misleading experimental data due to dropped packets, and shaped a reward function to be properly scaled for convergence. Furthermore, we explored the MPC's performance at different control frequencies. Though lower control frequencies had higher predictive power, it came at a cost of lower control authority, and this tradeoff was explored quantitatively at control frequencies of 25, 50, and 100 Hz. With our method, we most successfully achieved quadrotor hovering of up to 6 seconds from fewer than 10 minutes of collected data.

To further investigate learning algorithms for low level sensing and control, we also employed model-free learning techniques using the MPC dynamics model to roll out policies. The model-free algorithms utilized were double Q-learning, deep deterministic policy gradient, soft actor-critic, and twin soft actor-critic. In simulation, we found that soft actor-critic algorithms yielded the highest returns of all discrete and continuous action, though high return variance occurred for all algorithms due to compounded errors in an imperfect dynamics model. Though transferring simulated policies to the real world was unsuccessful, policies trained on less expressive state and action spaces demonstrated more reasonable reactions to Euler angle disturbances on the real quadrotor. On the other hand, more expressive policies yielded significantly better results in simulation, but only because it exploited actions that were infeasible on the real system. Furthermore, we conducted initial experiments with synthesizing a PID controller with our neural net dynamics model that would match the parameters on the Crazyflie built-in controller, and we wish to improve and automate this generation via Bayesian optimization.

Physical challenges with using the Crazyflie platform are also presented in this work. We quantified the shift of PWM equilibrium values of the motors after a strong collision of over 10% and measured the increasing sensor noise degradation over the lifetime of the quadrotor. Moreover, a detailed analysis of the effect of battery voltage on state predictions was conducted, and it was found that including battery voltage in the dynamics model explicitly was superfluous. Specifically, our finding that there was an inverse relationship between battery voltage and motor thrust suggests that battery voltage was latent to other network variables.

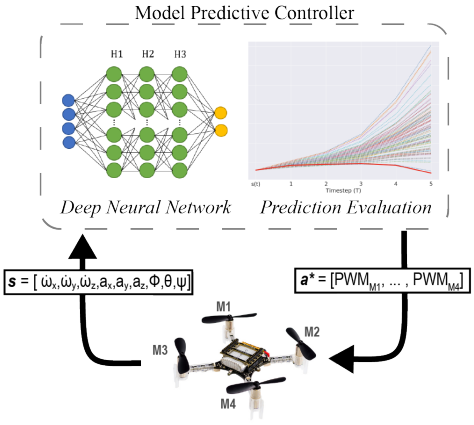

Fig. 1: Workflow diagram of the model predictive controller used to stabilize the Crazyflie during hovering. A trained dynamics model, using only raw sensor input, is used with a random shooting MPC to output low level PWM motor values at a set control frequency.

Low Level Control of a Quadrotor with Deep Model-Based Reinforcement Learning

Nathan O. Lambert and Isabella Huang

***Abstract*— This work demonstrates the first use of model-based reinforcement learning for stabilizing, low-level control of a quadrotor at less than 50Hz using only on-board sensors, with no initial system knowledge. Our approach uses a general algorithm for learning a low-level (i.e. direct motor input signals) controller, whereas comparable robot controllers are typically hand engineered and tuned. We show that the full radio communication, model prediction, and control pipeline can function at frequencies greater than 150Hz. We detail the firmware modifications required for radio communication of state data and raw PWM commands, as well as the implementation of a relatively high-frequency external model predictive controller. Our dynamics model is a neural network tuned to predict the Euler angles, linear accelerations, and angular accelerations at the next time step, with a regularization term on the variance of predictions. The model predictive controller, sampling uniformly from bounded action spaces around equilibrium, transmits best actions from a GPU-enabled ROS base-station to the quadrotor firmware via radio. The quadrotor achieved hovering capability of up to 6 seconds from fewer than 7 minutes of fully experimental (either on-policy or random action) training data. Lastly, model-free algorithms were explored as well. Although the transfer problem from simulation to real world flights was not addressed in this work, the interplay between policy expressiveness and limited experimental data was explored in both simulation and tests on the real system.**

I. Introduction

The design of appropriate robot controllers is traditionally a time-consuming and expert-based process. Even for relatively simple systems such as a quadrotor, tuning the parameters of the low-level PID controller can be challenging and often results in sub-optimal controllers. Control frequency and tuning becomes even more critical as the vehicle size decreases due to the subsequent change in the highly non-linear dynamics of the system. In this paper, we investigate the question: Is it possible to autonomously learn competitive low-level controllers for a quadrotor from scratch, i.e., without bootstrapping or demonstration? To answer this we turn to model-based reinforcement learning (MBRL) — a compelling approach to synthesize controllers even for systems without analytic dynamics models and with high cost per experiment. MBRL has been shown to operate in a data-efficient manner to control robotic systems by iteratively learning a dynamics model and subsequently leveraging it to design controllers [1], [2]. Building on the dynamics model, we further investigate the capabilities of offline dynamics models trained on experimental data to generate control policies from model free algorithms such as TSAC [3] and DDPG [4] or parameter regression for traditional PID controllers. Although model-free algorithms have successfully been used in conjunction with model-based algorithms to tackle unpredictable noise dynamics, we investigate the efficacy of model-free algorithms as standalone controllers in this work.

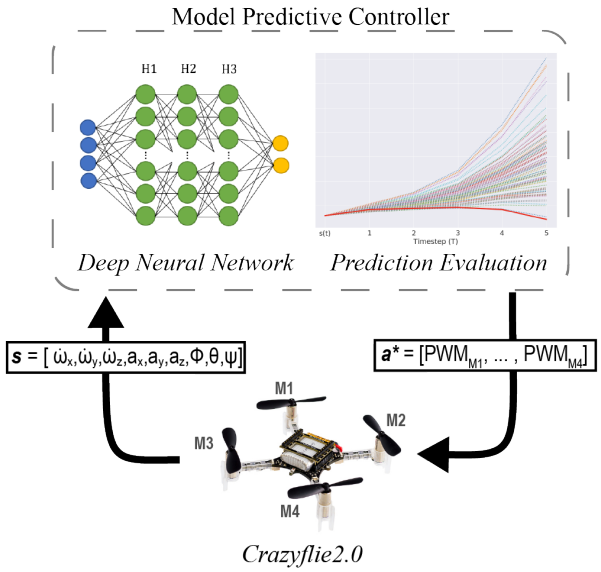

Fig. 2: The closed loop model predictive controller used to stabilize the Crazyflie. Using this MBRL architecture, we achieve stable hovering with only 5,000 trained datapoints – equivalent to 100 s of flight.

Our MBRL solution employs deep neural networks to learn a forwards dynamics model, coupled with a 'random shooter' model predictive controller (MPC) — see Figure 2 — similar to the one evaluated in simulation by [5]. This approach can be efficiently parallelized on a graphic processing unit to achieve low-level, real-time control at high frequency. Using this approach, we demonstrate on the Crazyflie (an established commercial quadrotor often used for research) the controlled hover of a quadrotor directly from raw sensor measurements and application of pulse width modulation (PWM) motor signals. Repeated stable hover of around two seconds is achieved with fewer than 10 minutes of fully-autonomous training data, demonstrating the ability of MBRL to control robotic systems in the absence of: any a priori knowledge of dynamics; any pre-configured internal controllers for stability or actuator response smoothing; and any expert demonstration. The comparison of different learning rates and control capability at 25 Hz and 50 Hz indicate how changes in prediction time-frame impact performance.

The initial results in model-free policy learning shows the challenge of generating policies with an imperfect dynamics model, but hint at future possibilities. The paper concludes with exploration into many limitations of the Crazyflie platform including battery voltage measurements and sensor lifetime.

II. Related Works

A. Attitude and Hover Control of Quadrotors

Classical controllers (e.g., PID) in conjunction with theoretically derived models for the rigid body dynamics of the quadrotor are sufficient to control vehicle attitude [6]. Linearized models are sufficient to simultaneously control for global trajectory, stable hover, and attitude setpoint using well-tuned nested PID controllers [7].

Research focusing on developing novel low-level attitude and hover controllers has shown functionality in extreme nonlinear cases, such as for quadrotors with a missing propeller [8], with multiple damaged propellers [9], or with the capability to dynamically tilt its propellers [10]. Optimal control schemes have demonstrated results on standard quadrotors with extreme precision and robustness [11].

Our work differs by specifically demonstrating the possibility of attitude and hover control via real-time external MPC. Unlike other work on real-time MPC for quadrotors [12], [13], ours uses a dynamics model derived fully from in-flight data (i.e., with no a priori structure or terms) that takes motors signals as direct inputs. Effectively, our model encompasses only the actual dynamics of the system, while other implementations also incorporate the dynamics of previously existing internal controllers as well.

B. Learning for Quadrotors

Although learning-based approaches have been widely applied for trajectory control of quadrotors, implementations typically rely on sending controller outputs as setpoints to stable on-board attitude and thrust controllers. Iterative learning control (ILC) approaches [14], [15] have demonstrated robust control of quadrotor flight trajectories but require higher frequency on-board controllers for attitude setpoints. Learning-based model predictive control implementations which successfully track trajectories also wrap their control around on-board attitude controllers by directly sending Euler angle or thrust commands [16], [17]. Gaussian process-based automatic tuning of position controller gains has been demonstrated [18], but only in parallel with on-board motor thrust and attitude controllers tuned separately.

Model-free reinforcement learning has been shown to generate control policies for quadrotors that out-perform linear MPC controllers [19]. Although similarly motivated by a desire to generate a control policy acting directly on actuator inputs, this work used an external vision system for state error correction, operated with an internal motor speed controller enabled (i.e., thrusts were commanded and not motor voltages), and generated a large fraction of its data in simulated rollouts.

Machine learning techniques have also been widely applied to the problem of system identification for quadrotors. Bansal et al. used neural network models of the Crazyflie's dynamics to plan trajectories [20]. Our implementation differs by directly predicting change in attitude (e.g. body pose, angular rates) based on raw actuator signals.

Bayesian optimization has been applied to learn a linearized quadrotor dynamics model for tuning of an optimal control scheme [21]. While this approach is data-efficient and is shown to outperform analytic models, the model learned is task-dependent.

C. Model-based Reinforcement Learning

Functionality of MBRL has been demonstrated in simulation for multiple tasks in low data regimes, including quadrupeds [22] and manipulation tasks [23]. Low-level MBRL control (i.e., with direct motor input signals) of an RC car has been demonstrated experimentally, but the system is of much lower dimensionality and has passive stability [24]. Relatively low-level control (i.e., mostly thrust commands only passed through an internal governor before conversion to motor signals) of an autonomous helicoptor has been demonstrated, but required a ground-based vision system for error correction in state estimates as well as expert demonstration for model training.

Properly optimized neural networks trained on experimental data have shown test error below common analytic dynamics models for flying vehicles, but the models did not include direct actuator signals and did not include experimental validation through controller implementation [25]. A model predictive path integral (MPPI) controller using a learned neural network demonstrated data-efficient trajectory control of a quadrotor, but results were only shown in simulation and required the network to be bootstrapped with 30 minutes of control demonstration [2].

MBRL with trajectory sampling for control has been shown to outperform, in terms of samples needed for convergence, the asymptotic performance of recent model free algorithms in low dimensional tasks [5]. Our work builds on strategies presented, with most influence derived from work on "probabilistic" neural networks, to demonstrate functionality in an experimental setting — i.e., in the presence of real-world higher order effects, variability, and time constraints).

Neural network-based learned dynamics models with model predictive control have been demonstrated to function for experimental control of an under-actuated hexapod [26]. The hexapod platform does not have the same requirements on frequency or control error due to its passive stability, and incorporates a GPS unit for relatively low-noise state measurements. Our work has a similar architecture, but has improvements in the network model and model predictive controller to allow substantially higher control frequencies with noisy state data. By demonstrating functionality without global positioning data, the procedure can be extended to more robot platforms where only internal state and actuator commands are available to create a dynamics model and control policy.

D. Model Free Learning for Quadrotors

The advantage of model-free control is that the complex non-linear dynamics and uncertainties of a real system do not have to be modelled perfectly. As such, model-free methods have proven to be effective for robust control when disturbances exist and computational resources are limited [27]. Model-free methods for learning local parameters was used to augment an LQR feedback controller in [28]. This method outperformed the pure LQR controller, even when the LQR parameters were deliberately degraded.

It has also been shown in simulation that a model-free based terminal sliding-mode control system, which combines a model-free policy with sliding-mode control techniques to eliminate bounded tracking errors in finite time, outperforms control methods like PID, iterative PID, backstepping, and pure sliding-mode [29]. Augmenting classical controllers with model-free estimators extends to even more variants, such as time-delay estimation-based control [30], piecewise continuous recursive control [31], and algebraic methods-based adaptive PID control [32]. However, in none of these works was the control input at the motor-level, since the mapping between motor control and thrust were assumed to be known. For the scope of this project, we intend to explore the usage of different model-free algorithms on the raw sensor data without using any explicit quadrotor dynamics model in conjunction. Our motivation for exploring this regime is to inform future processes of developing control laws for mechanical systems with even more subtle non-linearities and uncertainties.

III. Hardware System

A. ROS System

We used the open-source Crazyflie 2.0 quadrotor as our experimental hardware platform. The Crazyflie is $27\,\mathrm{g}$ and $9\,\mathrm{cm}^2$, so the rapid system dynamics create a need for a high speed controller; by default, the internal PID controller used for attitude control runs at $500\,\mathrm{Hz}$, with Euler angle state estimation updates at $1\,\mathrm{kHz}$. This section specifies the system involved in controlling the quadrotor and the firmware modifications required for external stability control.

All components we used are based on publicly available and open source projects. We used the Crazyflie ROS interface supported here: github.com/whoenig/crazyflie_ros. This interface allows for easy modification of the radio communication and employment of the learning framework. Our ROS structure is simple, with a Crazyflie subscribing to PWM values generated by a controller node, which processes radio packets sent from the quadrotor in order to pass state variables to the model predictive controller (as shown in Figure 3). The Crazyradio PA USB radio is used to send commands from the ROS server; software settings in the included client increase the maximum data transmission bitrate up to 2Mbps and a Crazyflie firmware modification improves the maximum traffic rate from $100\,\mathrm{Hz}$ to $300\,\mathrm{Hz}$.

In packaged radio transmissions from the ROS server we define actions directly as the pulse-width modulation (PWM) signals sent to the motors. To assign these PWM values directly to the motors we bypass the controller updates in the standard Crazyflie firmware by changing the motor power distribution whenever a CRTP Commander packet is received (see Figure 3) instead of modifying a controller setpoint value. The Crazyflie ROS package sends empty ping packets to the Crazyflie to ask for logging data in the returning acknowledgment packet; without decreasing the logging payload and rate we could not simultaneously transmit PWM commands at the desired frequency due to radio communication constraints. We created a new internal logging block of compressed IMU data and Euler angle measurements to decrease the required bitrate for logging state information, trading state measurement precision for update frequency. Action commands and logged state data are communicated asynchronously; the ROS server control loop has a frequency set by the ROS rate command, while state data is logged based on a separate ROS topic frequency. To verify control frequency and reconstruct state action pairs during autonomous rollouts we use a round-trip packet ID system. The controller will not send another command until the Crazyflie logging confirms that the PWM ID from the (at most two) prior MPC update has been set to the motors. This check confirms that a radio delay or buffer growth will not propagate through the system.

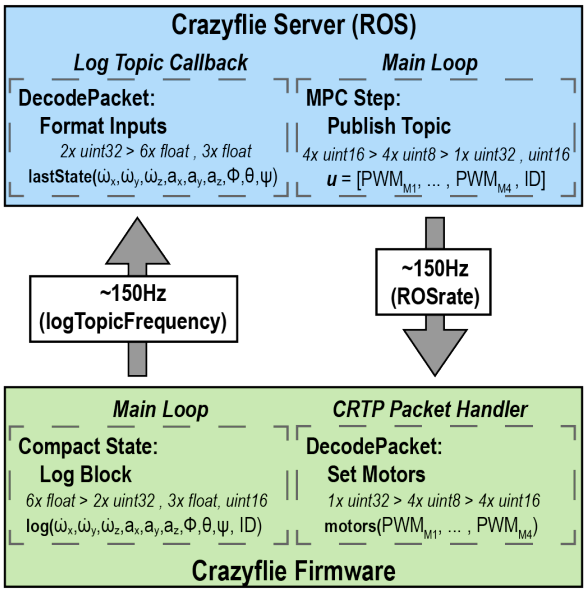

Fig. 3: The ROS system and the Crazyflie communicate in an minimal fashion. The ROS side computer passes control signals and state data between the model predictive controller node and the Crazyflie ROS server. The Crazyflie server packages Tx PWM values to send and unpacks Rx compressed log data from the robot.

Our firmware code will be made publicly available upon publication at github.com/natolambert/crazyflie-firmware-pwm-control.

IV. Dynamics Model

Generating a dynamics model for the robot requires training a neutral network to fit a parametric function f_θ to predict the next state of the robot as a discrete change in state $s_{t+1} = s_t + f_\theta(s_t, a_t)$ or a raw next state as $s_{t+1} = f_\theta(s_t, a_t)$. In training, using a probabilistic loss function with a penalty term on the variance of estimates better clusters predictions for more stable predictions across multiple time-steps [5].

The probabilistic loss function assisted model convergence and the variance penalty helps maintain stable predictions on longer time horizons. Our networks are trained for 60 epochs with the Adam optimizer [33] with a learning rate of .0005 and a batch size of 32. The network design is summarized in Figure 4. All layers except for the output layer use the Swish activation function [34] with parameter $\beta = 1$.

Training a probabilistic neural network to approximate the dynamics model requires pruning of logged data (e.g. dropped packets) and scaling of variables to assist model convergence. Our state model is the vector of Euler angles (i.e., yaw, pitch, and roll or ϕ, θ, ψ), linear accelerations $(\ddot{x}, \ddot{y}, \ddot{z})$, and angular accelerations $(\dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z)$, as in Equation (1). The Euler angles are from the Crazyflie internal complementary filter algorithm while the linear and angular accelerations are measured directly from the on-board MPU-9250 9-axis IMU.

$$s_t = \left[\dot{\omega}_x,\ \dot{\omega}_y,\ \dot{\omega}_z,\ \phi,\ \theta,\ \psi,\ \ddot{x},\ \ddot{y},\ \ddot{z}\right]^T \tag{1}$$

We combine the state data with the four PWM values, $a_t = [m_1, m_2, m_3, m_4]^T$, to get the system information at time t. When fitting a model to prediction accelerations across time horizons greater than 5, the large steps in predictions can cause instabilities. To mitigate compounding errors, we predict the angular and linear accelerations from a Gaussian distribution, while the Euler angles are predicting a change in state conditioned on all the inputs. This change does not have a substantial effect on one time step predictions, but gave increased stability for offline predictions and longer

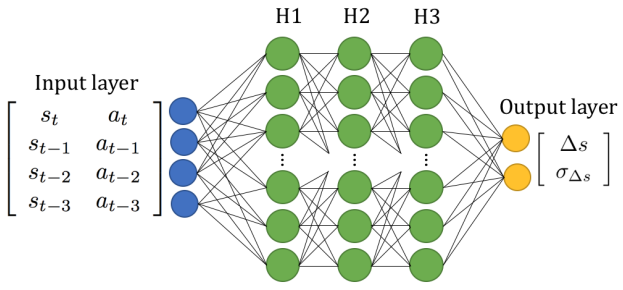

Fig. 4: The neural network dynamics model consists of the past 4 state-action pairs predicting the mean and variance of the change in state. The input is of dimension 52, predicting an 18 dimensional output. We use 3 hidden layers of width 500, with the Swish activation function. Training uses the Adam optimizer for 60 epochs with an initial learning rate of .0005, with a learning rate scheduler of .5 scaling every 20 epochs, and a batch size of 32.

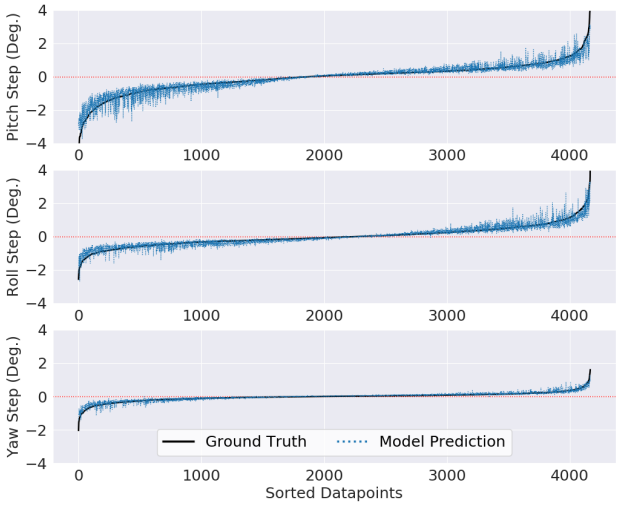

Fig. 5: One step predictions for change in Euler angles of the final model across the state space of data collected on the random action policy. The normalized mean squared errors from ground truth to the prediction for each of the Euler angles are Pitch: 0.130, Roll: 0.076, and Yaw: 0.032.

time horizons in the MPC. The neural networks are cross-validated to confirm using all state data (i.e., including the relatively noisy raw gyroscope and accelerometer measurements) improves prediction accuracy in the change in state. The ones step predictions for our final model is shown in Figure 5.

While the dynamics for a quadrotor are often represented as a linear system of equations, for a MAV at high control frequencies motor step response and thrust asymmetry heavily impact the change in state, resulting in a heavily nonlinear dynamics model. The step response of a Crazyflie motor from PWM 0 to max has been shown to be $250\,\mathrm{ms}$, so our update time-step of $6.7\,\mathrm{ms}$ is short enough for motor spin-up to contribute significantly to learned dynamics. To account for spin-up, past system information is appended to the current state and PWMs to generate an input into the neural network model that includes past time. From the exponential step response and with a bounded possible PWM value within $p_{eq} \pm 5000$, the motors need approximately $25\,\mathrm{ms}$ to reach the desired rotor speed; when operating at $150\,\mathrm{Hz}$, the time step between updates is $6.7\,\mathrm{ms}$, leading us to an appended state and PWM history of length 4. This state action history length was validated as having the lowest test error on our data-set (lengths 1 to 10 evaluated). This yields the final input to our neural network, ξ, being of length 52, with 4 state and action vectors concatenated:

$$\xi_t = \left[s_t\ s_{t-1}\ s_{t-2}\ s_{t-3}\ a_t\ a_{t-1}\ a_{t-2}\ a_{t-3}\right]^T \tag{2}$$

V. Model Based Control

Model predictive control provides a framework for evaluating many action candidates using a given dynamics model. We employ a 'random shooter' MPC, where a set of N randomly generated actions are simulated over a time horizon T. The best action is decided by a user designed objective

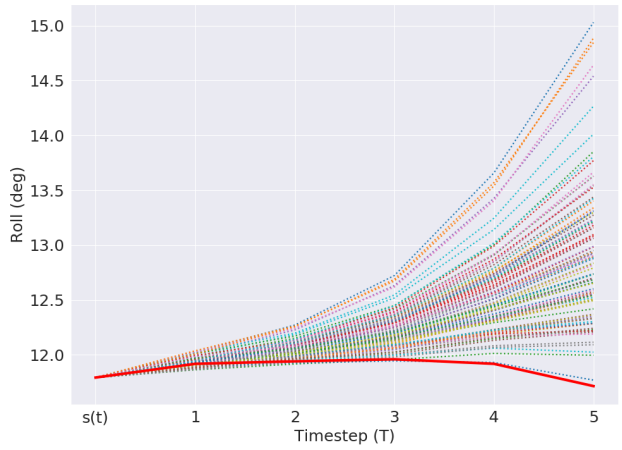

Fig. 6: Each line is a predicted future state from a candidate action, with the chosen "best action" highlighted in red. The candidate actions rapidly diverge around timesteps $3-5$, and continue to do so further into the future. The cost function is designed so the actions with extreme responses are eliminated by the MPC, which then return a more reasonable action from the center of the distribution.

function that takes in the simulated trajectories $\hat{X}(a, s_t)$ and returns a best action, a^*; as visualized in Figure 6. The candidate actions are 4-tuples of motor PWM values centered around the stable hover-point for the tested Crazyflie. The 4 PWM values in one sample action $a_i = (a_{i,1}, a_{i,2}, a_{i,3}, a_{i,4})$, where the index $i = 1, 2, \cdots N$. The candidate actions are constant across the time horizon T. For a single sample a_i, each $a_{i,j}$ is chosen from a uniform random variable on the interval $[p_{eq,j} - \sigma, p_{eq,j} + \sigma]$, where $p_{eq,j}$ is the equilibrium PWM value for motor j. The range of the uniform distribution is controlled by the tuned parameter σ; this has the effect of restricting the variety of actions the Crazyflie can take. For the given range of PWM values for each motor, $[p_{eq} - \sigma, p_{eq} + \sigma]$, we discretize the candidate PWM values to a step size of 256 to match the future compression into a radio packet. This discretization of available action choices also increases the coverage of the 4- dimensional candidate action space without an increase in N that would lead to decreased control frequency.

The objective function we designed for stability seeks to minimize distance from the origin for pitch and roll, while adding additional cost terms to Euler angle rates. The omission of global yaw term is because yaw does not reset to 0 at each takeoff and seeking a yaw of zero is not necessary for stability.

$$\begin{aligned} a^* &= \arg\min_a \; c\big(\hat{X}(a, s_t)\big) \\ &= \arg\min_a \sum_{t=1}^{T} \lambda_\theta(\psi_t^2 + \theta_t^2) \\ &\qquad + \lambda_\nabla(\dot{\psi}_t^2 + \dot{\theta}_t^2 + \dot{\phi}_t^2) \end{aligned} \tag{3}$$

In this objective function, λ_θ maps roughly to a proportional, or gain, term and λ_∇ corresponds to a derivative, or damping, term.

The MBRL approach was run at $25\,\mathrm{Hz}$ and $50\,\mathrm{Hz}$ to compare learning rate and flight performance at multiple frequencies. Both of the MPC's are operating on a longer time horizon of $T = 12$ with $N = 5000$. At $25\,\mathrm{Hz}$ this relates to a time prediction of $480\,\mathrm{ms}$ and at $50\,\mathrm{Hz}$ this is $240\,\mathrm{ms}$, so both are predicting further out than the motor rise time to a held action. This controller iteration is limited by control frequency to correct from minor disturbances or control errors from unmodeled states. When testing at $50\,\mathrm{Hz}$, about 20% of the flights after rollout 6 failed due to drift and collisions that were not avoidable with current sensors. The controller is capable of controlling at over $200\,\mathrm{Hz}$ with the current structure of dynamics models, but that will be limited to $T = 1$ on an Nvidia Titan Xp. Our control algorithm is summarized in Algorithm 1.

Early tests controlling at a frequency of $100\,\mathrm{Hz}$, $T = 6$, $N = 5000$ with a model trained on $50\,\mathrm{Hz}$ data was extremely promising, with over $\frac{2}{3}$ of flights failing due to drift, but further experiments on this were limited due to sensor degradation, as explored in IX-B and shown in Figure 18. We will explore further testing in this area as it indicates a potential to leverage the predictive power with a higher control frequency to mitigate minor disturbances.

Our controller code will be made publicly available upon publication at github.com/natolambert/crazyflie-ros-mbrl.

Algorithm 1 ROS MPC Summary

Gather random data $\mathbb{D}_0$, train initial model $f_{0,\theta}$
for Rollout $r = 1$ to R **do**
  while Flying **do**
    ROS Rx: (s_t, a_t)
    Compile $\xi_t = [s_t, a_t, \cdots, s_{t-3}, a_{t-3}]$
    for $i = 1$ to N **do**
      $\{a\}_{i,j} \sim U\big(p_{eq,j} - \sigma, p_{eq,j} + \sigma\big)$
      for $k = 1$ to T **do**
        $s_{t+k} = s_{t+k-1} + f_{i,\theta}(\xi_t)$
      end for
      $obj(a_i) = \sum_{k=1}^{T} c(s_{t+k}, a_i)$
    end for
    ROS Tx: $a^* = \arg\min_a obj(a_i)$
  end while
  Append data $\mathbb{D} = \mathbb{D}_r \cup \cdots \cup \mathbb{D}_0$, train $f_{r,\theta}$ on $\mathbb{D}$
end for

VI. Model Free Control

To test the efficacy of model-free algorithms in controlling the quadrotor, we trained several different policies based on data collected from real flights. The goal was to investigate the most promising algorithms on which to based model-free policies in this setting. To this end, we designed a custom OpenAI Gym environment and performed all experiments at 50 Hz. An observation in the environment at time t is defined to be $(s_t, s_{t-1}, s_{t-2}, a_{t-1}, a_{t-2})$, and the policy should output the best action a_t. The bounds of a valid observation

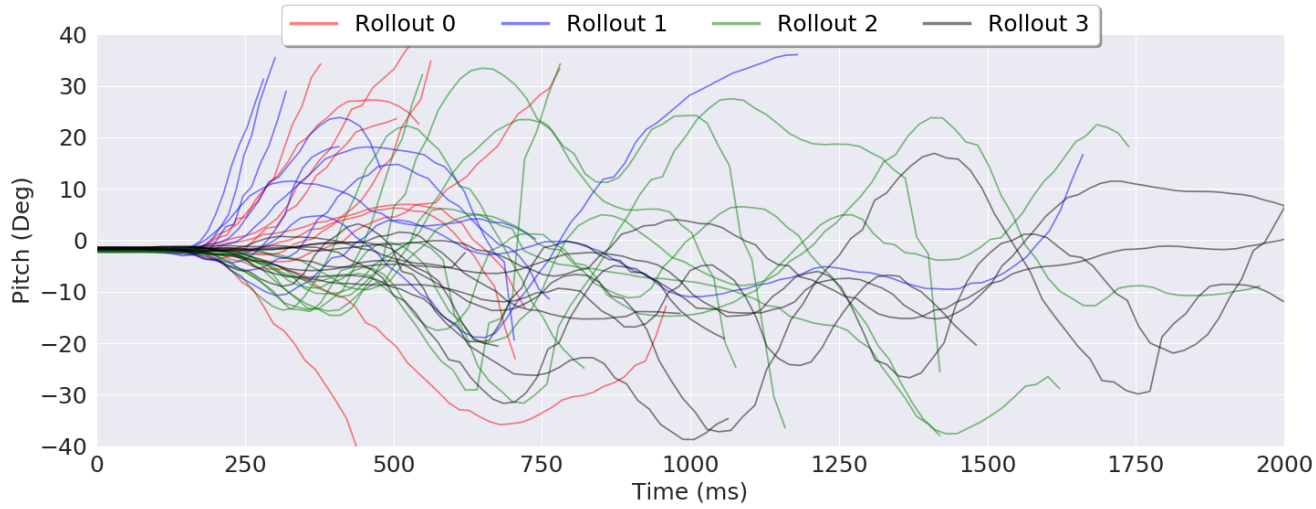

Fig. 7: The pitch over time for each flight in the first four rollouts of MBRL learning at 50 Hz, showing the rapid increase in control ability on limited data. The random and first controlled rollout show little ability to correct, but rollout 3 is already progressing multiple flights beyond 2 seconds.

were defined based on the numerical state and action bounds observed in real flight data in order to maintain fidelity with the physical plant. To preserve the latent dynamics in the stacked states of each observation, recent states were drawn from real observed data with added Gaussian noise, and the environment dynamics were governed by the dynamics models trained for MPC with 9000 datapoints. Furthermore, the following reward function was defined similarly as in the MPC case, except an additional failure penalty $\lambda_{fail}$was found to lead to better converging policies:

$$r(s_{t-1}, a_{t_1}) = \begin{cases} -\lambda_{fail} & \text{if } s_t \text{ failed} \\ r_{\theta,t} + r_{\nabla,t} & \text{otherwise} \end{cases} \quad (4)$$

where

$$r_{\theta,t} = \max_{data} \left(\lambda_\theta(\psi^2 + \theta^2)\right) - \lambda_\theta(\psi_t^2 + \theta_t^2) \quad (5)$$

is higher for lower pitch and roll angles, and

$$r_{\nabla,t} = \max_{data} \left(\lambda_\nabla(\dot{\psi}_t^2 + \dot{\theta}_t^2 + \dot{\phi}_t^2)\right) - \lambda_\nabla(\dot{\psi}_t^2 + \dot{\theta}_t^2 + \dot{\phi}_t^2) \quad (6)$$

is higher for lower angular accelerations. The failure condition of s_t occurred when either its pitch or roll angles is greater than 30 degrees.

In selecting model-free algorithms, one approach was to discretize the action space and employ double deep Q-learning [35]. In this regime, each motor j was allowed to take one of three values in the set $\{p_{eq,j} - \sigma, p_{eq,j}, p_{eq,j} + \sigma\}$ for a total of 81 unique combined motor actions. The σ value was selected to be 5000 to adequately cover the range of PWM values required for hovering using MPC, and to avoid supplying candidate actions where the dynamics model has no knowledge. On the other hand, we also investigated policies over continuous action spaces including deep deterministic policy gradients (DDPG) [4], soft actor-critic (SAC) [36], and Twin SAC, a combination of SAC and twin dueling deep deterministic policy gradient (TD3) [3]. The benefit of exploring both discrete and continuous policies is to study the trade-off between desiring an expressive policy network while not having access to unlimited flight data to best train it. DDPG, a natural extension to DQN that also utilizes a replay buffer and separate target network, employs an actor-critic architecture to handle selecting over continuous actions. SAC attempts to maximize entropy in its policy, which evades the traditional Q-learning framework and implicitly embodies better exploration. Furthermore, TSAC builds upon SAC by using two critic networks to minimize overfitting, which is essential to our setting with an imperfect dynamics model. Fig. 8 details the test returns of these algorithms during training, with TSAC being superior amongst the other continuous action algorithms.

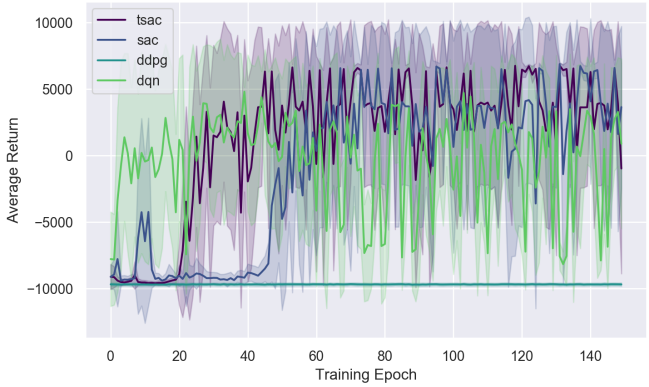

Fig. 8: Test returns for the four different model-free algorithms vs. training epoch. High variances appear to be an artifact of using an imperfect dynamics model whose error in estimation blows up when compounded.

Random rollouts of the policy for different initial conditions are plotted for the DQN and TSAC policies in Figures

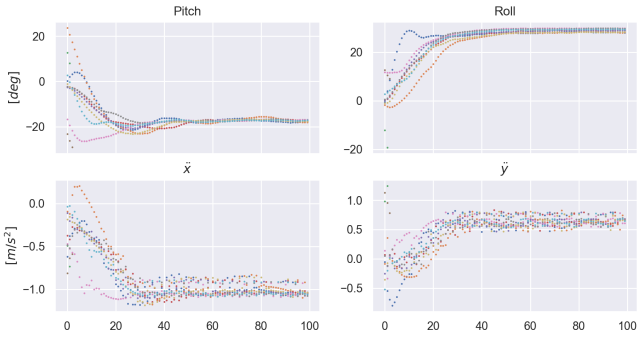

Fig. 9: Ten simulated rollouts with the DQN policy.

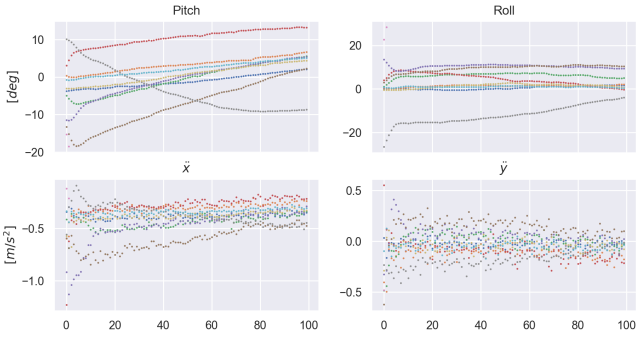

Fig. 10: Ten simulated rollouts with the TSAC policy.

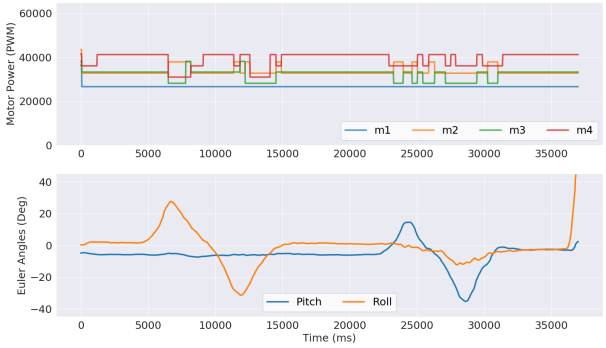

Fig. 11: Quadrotor PWM response to Euler angle disturbances with the DQN policy. The PWMs respond reasonably to changes in Euler angles, but not consistently enough for controlled flight.

Fig. 12: Quadrotor PWM response to Euler angle disturbances with the TSAC policy. The PWMs do not respond in a way to fly, but in an exploitative manner of the dynamics model to maximize reward.

9 and 10 respectively. We see that in the DQN policy, the quadrotor maintains a pitch and roll angle very close to the failure value of 30 degrees throughout its trajectory. To achieve this orientation, the linear accelerations are necessarily non-zero as evidenced in Fig. 9. Though undesirable for real-world testing where flight space is limited, this result is reasonable since the action set available in the discretized environment was not dense enough to fine-tune and steer the quadrotor pitch and roll angles to zero. On the other hand, Fig. 10 demonstrates that the TSAC policy is able to steer the pitch and roll angles much closer to zero and maintain a hovering status with minimal angular and linear accelerations.

As expected, since the transfer problem of importing a simulated policy to the real world was not addressed in this work, none of these model-free policies yielded successful hovering on a physical quadrotor. This is because model-free policies are trained to maximize rewards over longer horizons than in MPC, so errors in the dynamics model are compounded. However, the reactiveness of the quadrotor held in-hand to pitch and roll angle disturbances are shown in Figures 11 and 12. In the DQN case, the motors do react to angular disturbances, but the same cannot be said for the TSAC case, where the motor values are all lower than at their equilibrium values. It appears that the TSAC policy had exploited the imperfect dynamics model and converged on an unrealistic set of feasible actions for flight, where low motor thrust will keep the quadrotor on the ground following a brief hop from takeoff.

Overall, without more data to yield a better dynamics model, the use of an expressive environment with complexities such as stacked states is unlikely to produce reasonable actions in the real world, whereas more constrained states induced by DQN yielded a proper qualitative response. However, continuous policies are still promising methods for handling unmodeled noise for use in conjunction with a model-based controller rather than being standalone. An additional consideration for improving gradients of policies trained on neural network dynamics models is to constrain the gradients to less sensitive to exploding gradients [37].

VII. Other Control

The expressive power of a neural network dynamics model lends it's hand to use in other types of control. The neural network can be used as a plant in more traditional controllers [20] or as an imitative copy of expressive, computationally intensive control. An notable baseline to compare the performance of the MBRL solution to is that of generated PID or standard controllers. Recent work showed the capability of Bayesian optimization for synthesizing PID parameters for experimental robots [38]. Exploration of the synergy between Bayesian optimization for parameter tuning with an expressive neural network dynamics model has potentially to uncover other viable solutions for control of novel robots with a lower computational footprint. Initial experiments

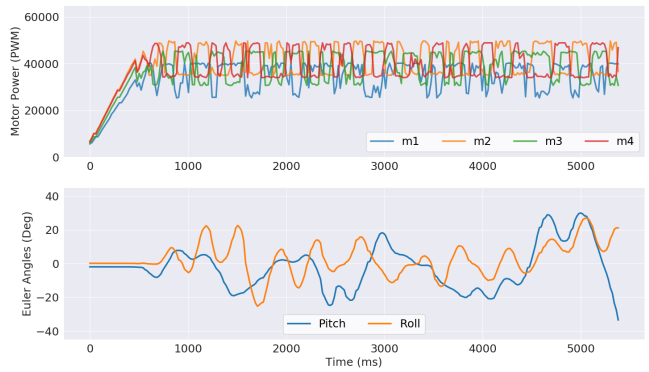

Fig. 13: The performance of the MPC algorithm over a test flight. The time under control for this flight is over 5 s. Above: The open-loop takeoff followed by the controlled PWM values over time. Below: Controlled pitch and roll.

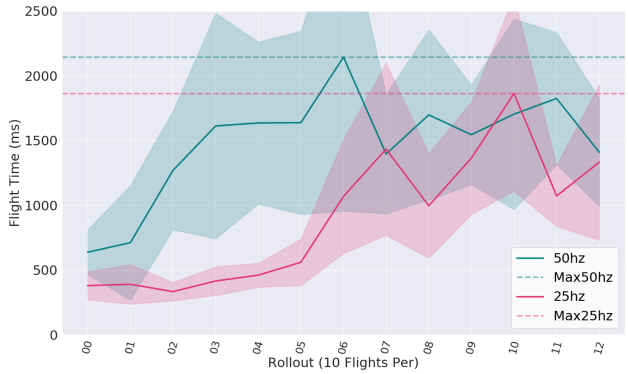

Fig. 14: Mean and standard deviation of the 10 flights during each flight for MBRL at 25 Hz and 50 Hz. The 50 Hz shows a slight edge on final performance, but a much quicker learning ability per flight by having more PWM switches during control.

Fig. 15: Mean flight time of each MBRL rollout plotted verses the logarithm of the number of available points at train time for each model. The higher control frequency allows the controller to learn faster on wall time, but the plot indicates that there is not a notable difference between control ability when the number of trained points are equal. There is a continuing upward trend of flight time verses training points, but it is difficult to obtain more data in experiment.

showed the ability to close a PID feedback loop matching the Crazyflie's design around a neural net dynamics model, but work is needed to automated the generation of the PID parameters.

VIII. Evaluation

The performance of our controller is measured by the average flight length over each roll-out. This section is discussing the performance of the MBRL solutions using a MPC, as the model-free solutions are yet to demonstrate any controlled flight. An example flight with clear controlled Euler angles is shown in Figure 13. Failure is primarily due to collision with an object due to drift, or, as in many eariler roll-outs, when flights reach a pitch or roll angle over $40°$. In both cases, an emergency stop command is automatically sent to the motors to prevent quadrotor damage during data collection. Along with the collisions due to drift, the simple onboard state estimator shows heavy drift on the Euler angles following a rapid throttle ramping; both are limiting factors on the length of controlled flight without a dead reckoning system.

A. Learning Process

The learning process follows the reinforcement learning framework of collecting data and iteratively updating the policy. We trained an initial model f_0 on 124 and 394 points of dynamics data at 25 Hz and 50 Hz, respectively, from the Crazyflie being flown by a random action controller supplying PWM values. Starting with this initial model as the MPC plant, the Crazyflie undertakes a series of autonomous flights from the ground with a 250 ms ramp up, open-loop takeoff followed by on-policy control while logging data to the ROS base-station. Each rollout is a series of 10 flights, which causes large variances in flight time. The initial roll-outs have less control authority and inherently explore more unstable state spaces, which is valuable to future iterations that wish to recover from higher pitch and or roll. The learning curves are showing in Figure 14. There is a trend between learned performance at both frequencies and the number of trained points for the model, as shown in Figure 15. The continuing upward trend between logarithmic points and flight time indicates further data collection could enhance flight performance.

B. Performance Summary

The initial learning results are shown in Figure 7, where the rapid visible improvement in results is evident. Both 25 Hz and 50 Hz control show a level off in performance after reaching a certain data threshold. The longest flights during these rollouts is over 5 s for both frequencies. The leveling off could be caused by a slow in improvement of the dynamics model, with orders of magnitude growth in amount of data potentially being needed for further performance improvements.

The best results over the model roll-outs are shown in Figure 7, where later roll-outs are clearly longer on average.

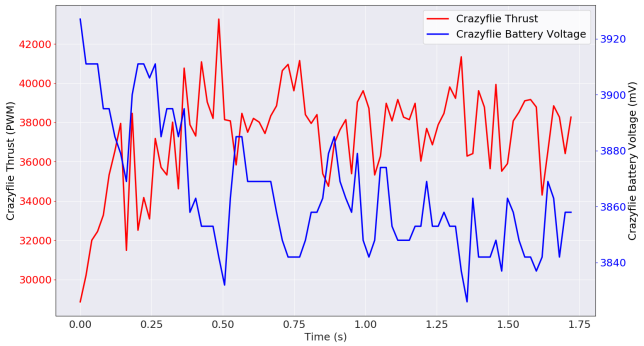

Fig. 16: The logged battery voltage and mean PWM of the 4 motors across a flight. There is a dominant relationship between the logged battery voltage and the current thrust.

The data used for training and the results of the roll-outs is summarized in Figure 14, which shows the average flight length double from the initial random policy to the final controller deployed after 1 random and 4 on-policy roll-outs. This controller demonstrated the ability to hover, following a "clean" open-loop takeoff, for multiple seconds. An example of a test flight is shown in Figure 13, where the response to pitch and roll error is visible.

The system has multiple limitations resulting in the short time-scale of flight. First, the PWM equilibrium values of the motors shift by over 10% following a strong collision, causing the true dynamics model to shift semi-randomly over time. Additionally, the internal state estimator does not track extreme changes in Euler angles accurately; our controller, which operates at a relatively low rate for motor control, needs carefully tuned damping terms to eliminate oscillations from open loop takeoff or control overshoot. Although relatively short hover times are achieved, we believe that overcoming the considerable system-level and dynamical limitations of controlling the Crazyflie in this manner showcase the expressive power of MBRL.

Current and future results will be updated here: https://sites.google.com/berkeley.edu/mbrl-quadrotor/

IX. Discussion

A. Battery Voltage Context

The Crazyflie has a short battery voltage of about 7 minutes of flight time and operation depends heavily on battery voltage, with it becoming uncontrollable on low voltages. When training models, we investigated logging battery voltage and adding it to the state passed to the neural network to improve prediction accuracy.

When operating the Crazyflie at control frequencies of greater than $100\,\text{Hz}$, the state dynamics become clearly biased at battery voltages less than 3,650 mV. The biased state dynamics can be seen in Figure 17, but the predictions do not improve when passing the battery voltage into the neural network dynamics model at any battery level. The RMS delta between a model trained with and without battery voltage is less than 1%.

A potential explanation for the lack of model improvement with logged battery voltage is that the current battery reading is latent in other variables past into the network, and the natural charge based variations in data are not dominant. The logged data shows a clear inverse relationship between battery voltage and current Crazyflie thrust, shown in Figure 16. This relationship is less likely to be apparent on quadrotors with separate motor voltage controllers, where the impedance of the motors changing with revolutions per minute would be compensated for.

B. Crazyflie Lifespan

Extended period of testing on individuals quadrotors demonstrated a finite lifetime. After many flights, performance would dip inexplicably. This is due to a combination of motor damage and or sensor decay. Motor damage causes a measureable change in the equilibrium PWMs for a given quadrotor. Figure 18 shows the change in the noise on the gyroscope before takeoff for all of the flights taken by the quadcopter used to collect data for this publication. The left two sections includes the data included in VIII, but the data taken at a control frequency of $75\,\text{Hz}$ was abandoned due to inconsistent performance. Some initial flights at $75\,\text{Hz}$ were extremely promising, but after a series of collisions via drift the quadrotor would not take off cleanly. Future work should investigate methods of mitigating the effect of sensor drift, potentially by conditioning the dynamics model on a sensor noise measurement or enforcing more safety constraints on flight.

X. Conclusion

This work is an exploration of the capabilities of model-based reinforcement learning for the low-level control of initially unknown systems. We demonstrate the firmware modifications, system design, and model learning considerations required to enable the use of a MBRL-based MPC system over radio at $150\,\text{Hz}$. We removed all robot-specific transforms and higher level commands (e.g. thrust and Euler angle rate setpoints) to only design the controller on top of a learned dynamics model to accomplish a simple task. The controller shows the capability to hover for multiple seconds at a time off of less than 7 minutes of collected data — approximately the full battery life flight time of a Crazyflie quadrotor. The successful hovering of a quadrotor underscores potential extensions of the presented MBRL framework for future low-level control.

There are multiple pathways to investigate for improving the performance of the current system as presented. Currently, the random shooter MPC method devotes a substantial amount of computation on action samples that will never produce a desirable state change. The controller could be improved by developing a prior weighting on the reward of each action that is developed over time, resulting in a weighted action sampling based on the current state. Another way to reduce computation could be to use a neural network policy trained to imitate the initial model predictive controller that would no longer require evaluating every

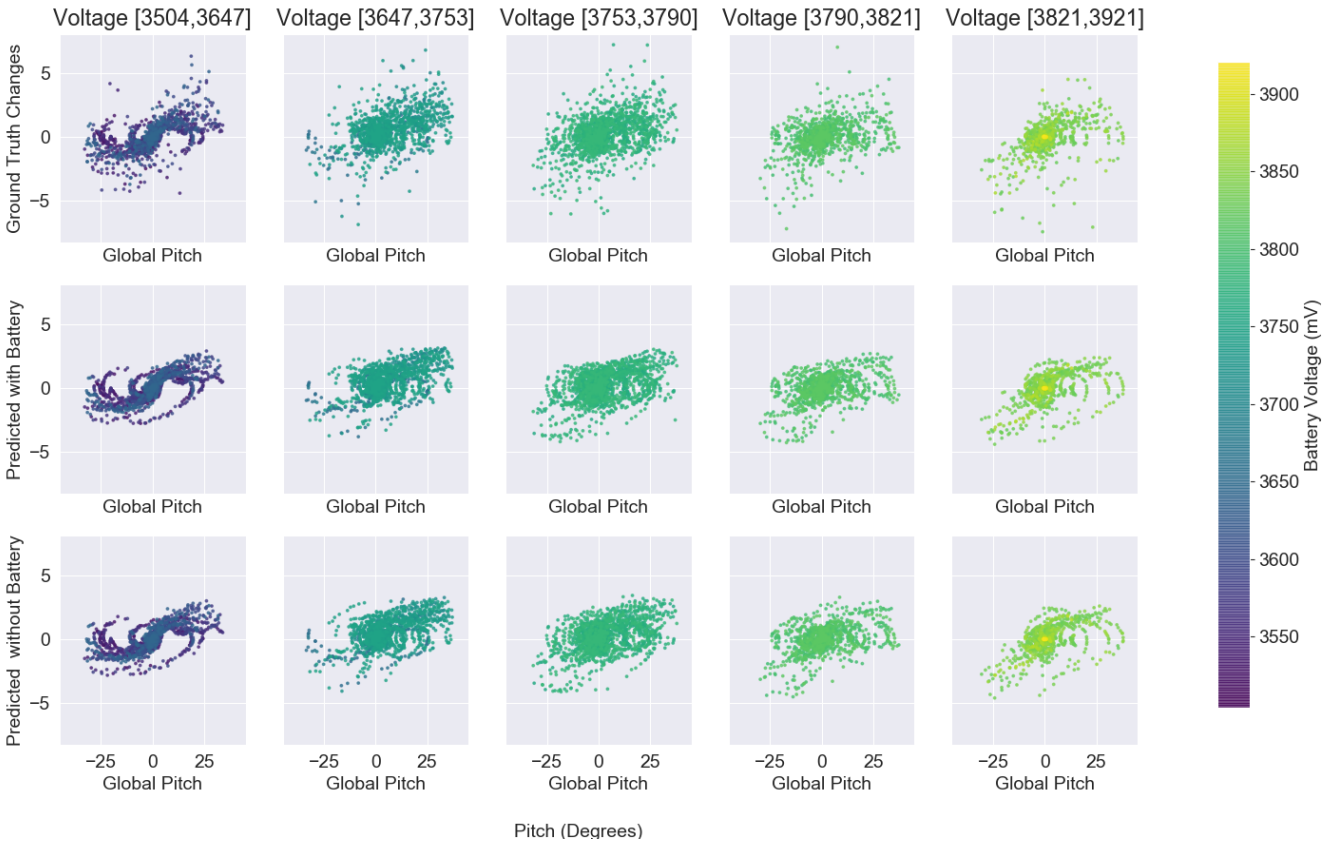

Fig. 17: This plot shows the effect of battery voltage on state predictions. The top row is the ground truth one step pitch changes, the middle is the predictions through a model trained with battery voltage included in the state, and the bottom is predictions without battery voltage included in the state. Both of the predictions show tighter grouping from the variance term on the probabilistic loss function, but there is an extremely low difference between the predictions with and the predictions without battery. The lack of difference in predictions indicates the battery voltage is latent to other variables passed into the network.

candidate action. Also, in our work, we maintained a general approach to the MPC block to avoid using a priori system knowledge; it is likely that performance could be improved with further controller specialization and objective function tuning. Combining the strides in improving the MPC with the model free and other controllers will allow us to expand the capabilities of the current system.

These initial results create opportunities for numerous experiments to improve and apply the MBRL system:

While in this work data is collected with human assistance when the robot crashes or drifts substantially, we envision a future approach that allows for data collection with no human intervention. Defining safety constraints within the model predictive controller, rather than just a safety kill-switch in firmware, could enable intelligent, safe exploration of state space, opening the door for fully autonomous learned control.

With similar robots, training can be run in parallel and the related dynamics models can be used as a weighted ensemble for prediction, hopefully decreasing total training time and improving performance. The ROS Crazyflie system is structured to easily interface with multiple robots, allowing a natural extension from safe state space exploration to multiple agents learning autonomously to fly together.

Direct synthesis of robot controllers operating on raw actuator inputs has exciting implications for development of novel robotic platforms. The emergent area of microrobotics combines the issues of under-characterized actuators and dynamics, weak or non-existent controllers, “fast” system dynamics and therefore instabilities, and extremely high cost-to-test [39]–[41]. Controller development based on data-efficient MBRL approaches could aid in the efforts to deploy autonomous millimeter-scale robots without significant time investment in controller design.

Moreover, our investigation into developing closed-form model-free policies indicated that although the policies were able to converge in simulation, it proved unsuccessful in the real world. This was largely due to imprecision in the trained dynamics model, and we expect these errors to be mitigated with more training data and better pruning. It would also be very interesting for future work to include leveraging MPC as a supervisor to train a closed-form policy, exploring trust-region policy optimization techniques [42], as well as using model-free policies to augment MPC rather than being standalone.

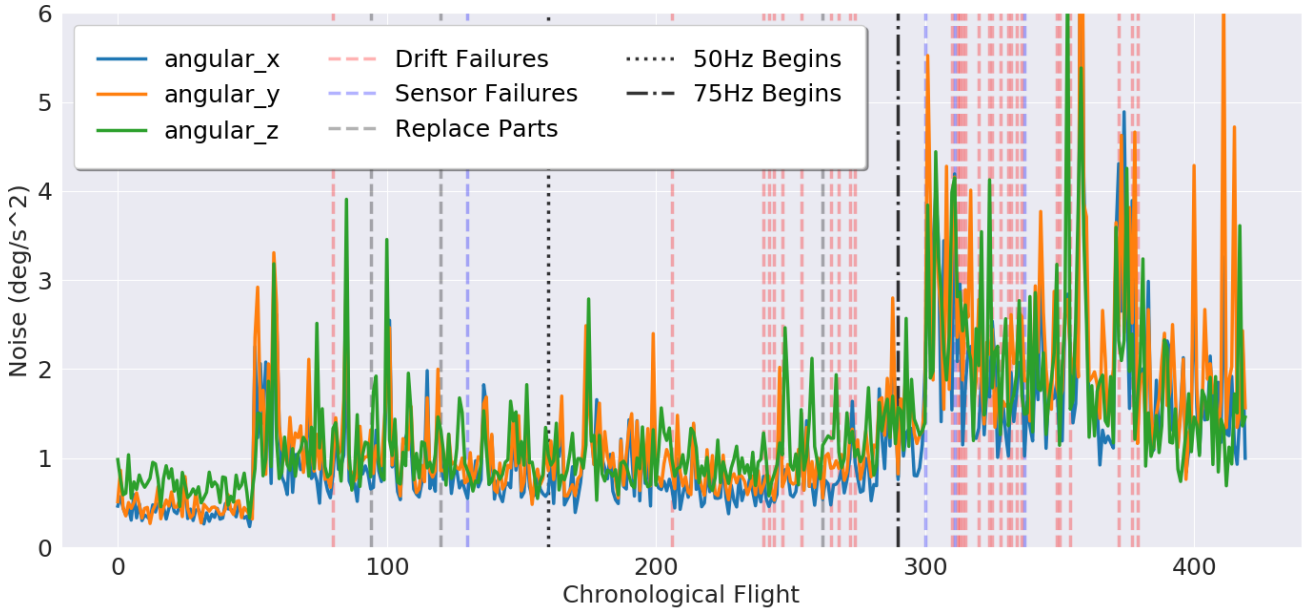

Fig. 18: The sensor noise on the 3 angular accelerations measured by the gyroscope of the MPU9250 of the Crazyflie before the robot takes off. The black vertical lines separate the rollouts at 25 Hz, 50 Hz and 75 Hz from left to right. The vertical lines indicate changes in hardware and collisions that would change the dynamics or state of the robot. The sensors clearly are subject to increasing noise over lifespan.

ACKNOWLEDGMENT

The authors thank the UC Berkeley Sensor & Actuator Center (BSAC), Berkeley DeepDrive, and Nvidia Inc. Nathan Lambert contributed a lot of the background and initial work in preparation for a paper submission. His work for this project consisted of extending the MBRL ideas, exploring traditional controllers, and analyzing Crazyflie lifetime and variables related to training dynamics models. Isabella contributed most of the model free development. The Git history will show code contributions. https://github.com/natolambert/dynamics-learn

REFERENCES

[1] M. Deisenroth, D. Fox, and C. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 37, no. 2, pp. 408–423, 2015.

[2] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *International Conference on Robotics and Automation (ICRA)*, 2017.

[3] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *CoRR*, vol. abs/1802.09477, 2018. [Online]. Available: http://arxiv.org/abs/1802.09477

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015. [Online]. Available: http://arxiv.org/abs/1509.02971

[5] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models," *Neural Information Processing Systems*, 2018.

[6] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles," *IEEE Robotics and Automation magazine*, vol. 20, no. 32, 2012.

[7] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[8] W. Zhang, M. W. Mueller, and R. D'Andrea, "A controllable flying vehicle with a single moving part," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3275–3281.

[9] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 45–52.

[10] M. Ryll, H. H. Bülthoff, and P. R. Giordano, "Modeling and control of a quadrotor uav with tilting propellers," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4606–4613.

[11] H. Liu, D. Li, J. Xi, and Y. Zhong, "Robust attitude controller design for miniature quadrotors," *International Journal of Robust and Nonlinear Control*, vol. 26, no. 4, pp. 681–696, 2016.

[12] M. Bangura, R. Mahony, *et al.*, "Real-time model predictive control for quadrotors," 2014.

[13] M. Abdolhosseini, Y. Zhang, and C. A. Rabbath, "An efficient model predictive control scheme for an unmanned quadrotor helicopter," *Journal of intelligent & robotic systems*, vol. 70, no. 1-4, pp. 27–38, 2013.

[14] A. P. Schoellig, F. L. Mueller, and R. DAndrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, no. 1-2, pp. 103–127, 2012.

[15] C. Sferrazza, M. Muehlebach, and R. D'Andrea, "Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control," in *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*. IEEE, 2017, pp. 5186–5192.

[16] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 279–284.

[17] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration and reinforcement learning," *arXiv preprint arXiv:1803.08287*, 2018.

[18] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller

optimization for quadrotors with gaussian processes," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 491–496.

[19] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a Quadrotor with Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[20] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning Quadrotor Dynamics Using Neural Network for Flight Control," *CDC*, no. 0931843, 2016.

[21] S. Bansal, R. Calandra, T. Xiao, S. Levine, and C. J. Tomiin, "Goal-driven dynamics learning via bayesian optimization," in *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*. IEEE, 2017, pp. 5168–5173.

[22] I. Clavera, A. Nagabandi, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt: Meta-learning for model-based control," *arXiv preprint arXiv:1803.11347*, 2018.

[23] A. Kupcsik, M. P. Deisenroth, J. Peters, A. P. Loh, P. Vadakkepat, and G. Neumann, "Model-based contextual policy search for data-efficient generalization of robot skills," *Artificial Intelligence*, vol. 247, pp. 415–439, 2017.

[24] P. Abbeel, *Apprenticeship learning and reinforcement learning with application to robotic control*. Stanford University, 2008.

[25] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3223–3230.

[26] A. Nagabandi, G. Yang, T. Asmar, G. Kahn, S. Levine, and R. S. Fearing, "Neural network dynamics models for control of under-actuated legged millirobots," *Intelligent Robots and Systems (IROS)*, 2018.

[27] J. Wang, M. Geamanu, A. Cela, H. Mounier, and S. Niculescu, "Event driven model free control of quadrotor," in *2013 IEEE International Conference on Control Applications (CCA)*, Aug 2013, pp. 722–727.

[28] Y. A. Younes, A. Drak, H. Noura, A. Rabhi, and A. E. Hajjaji, "Model-free control of a quadrotor vehicle," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 1126–1131.

[29] H. Wang, X. Ye, Y. Tian, G. Zheng, and N. Christov, "Model-freebased terminal smc of quadrotor attitude and position," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2519–2528, October 2016.

[30] K. Youcef-Toumi and T. A. Fuhlbrigge, "Application of decentralized time-delay controller to robot manipulators," in *Proceedings, 1989 International Conference on Robotics and Automation*, May 1989, pp. 1786–1791 vol.3.

[31] Y. Tian, H. Wang, and C. Vasseur, "Adaptive optimal trajectory tracking control of nonlinear affine in control system with unknown internal dynamics," in *Proceedings of the 33rd Chinese Control Conference*, July 2014, pp. 2234–2239.

[32] M. Fliess and C. Join, "Model-free control," *International Journal of Control*, vol. 86, no. 12, pp. 2228–2252, 2013. [Online]. Available: https://doi.org/10.1080/00207179.2013.810345

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941*, 2017.

[35] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[36] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018. [Online]. Available: http://arxiv.org/abs/1801.01290

[37] P. Parmas, C. E. Rasmussen, J. Peters, and K. Doya, "PIPPS: Flexible model-based policy search robust to the curse of chaos," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 4065–4074. [Online]. Available: http://proceedings.mlr.press/v80/parmas18a.html

[38] A. Doerr, D. Nguyen-Tuong, A. Marco, S. Schaal, and S. Trimpe, "Model-based policy search for automatic tuning of multivariate PID controllers," *CoRR*, vol. abs/1703.02899, 2017. [Online]. Available: http://arxiv.org/abs/1703.02899

[39] D. S. Drew, N. O. Lambert, C. B. Schindler, and K. S. Pister, "Toward controlled flight of the ionocraft: A flying microrobot using electrohydrodynamic thrust with onboard sensing and no moving parts," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2807–2813, 2018.

[40] D. S. Contreras, D. S. Drew, and K. S. Pister, "First steps of a millimeter-scale walking silicon robot," in *Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS), 2017 19th International Conference on*. IEEE, 2017, pp. 910–913.

[41] R. J. Wood, B. Finio, M. Karpelson, K. Ma, N. O. Pérez-Arancibia, P. S. Sreetharan, H. Tanaka, and J. P. Whitney, "Progress on picoair vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1292–1302, 2012.

[42] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," *CoRR*, vol. abs/1802.10592, 2018. [Online]. Available: http://arxiv.org/abs/1802.10592