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

Nathan O. Lambert[1,2], Daniel S. Drew[2], Joseph Yaconelli[3], Roberto Calandra[2], Sergey Levine[2], and Kristofer S. J. Pister[2]

***Abstract*— This work demonstrates the first use of model-based reinforcement learning for stabilizing, low-level control of a quadrotor 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 2 seconds from fewer than 7 minutes of fully experimental (either on-policy or random action) training data.**

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].

Our MBRL solution employs deep neural networks to learn a forwards dynamics model, coupled with a 'random shooter' model predictive controller (MPC) — see Figure 1 — similar to the one evaluated in simulation by [3]. 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.

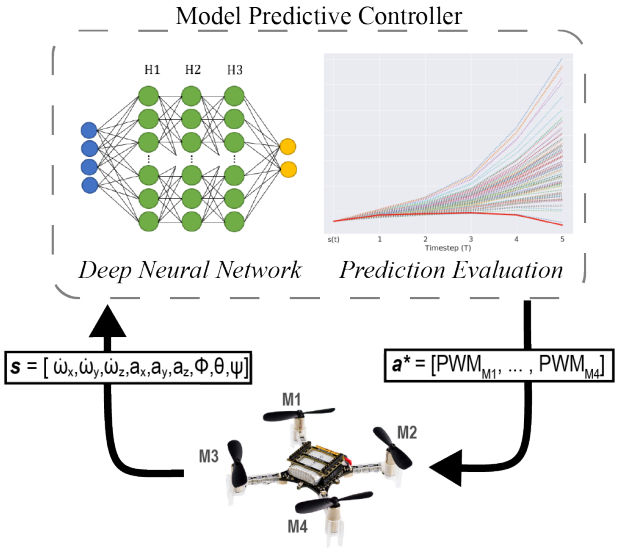

Fig. 1: The closed loop model predictive controller used to stabilize the Crazyflie. ROS holds the control loop of radio communication, model prediction, and action choice at 150 Hz. Using this MBRL architecture, we achieve stable hovering with only 55,000 trained datapoints – equivalent to 360 s of flight.

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.

II. Related Work

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 [4]. Linearized models are sufficient to simultaneously control

[1](Corresponding author: Nathan O. Lambert nol@berkeley.edu)
[2]Authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.
[3]Author is supported by the Berkeley SUPERB REU Program. Author is affiliated with Department of Computer Science, University of Oregon.

for global trajectory, stable hover, and attitude setpoint using well-tuned nested PID controllers [5].

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 [6], with multiple damaged propellers [7], or with the capability to dynamically tilt its propellers [8]. Optimal control schemes have demonstrated results on standard quadrotors with extreme precision and robustness [9].

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 [10], [11], 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 [12], [13] 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 [14], [15]. Gaussian process-based automatic tuning of position controller gains has been demonstrated [16], 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 [17]. 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 [18]. 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 [19]. 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 [20] and manipulation tasks [21]. 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 [22]. 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 [23]. 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 [3]. 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 [24]. 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.

III. Experimental Setup

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 $250\,\mathrm{Hz}$. 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.

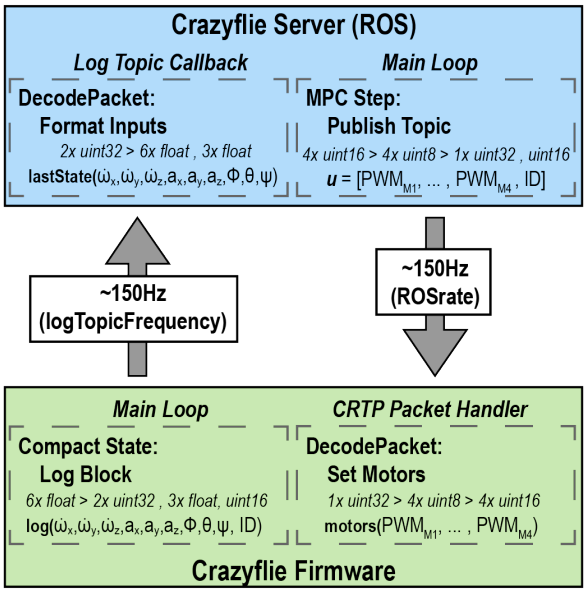

Fig. 2: The ROS system and the Crazyflie communicate in an minimal fashion. The ROS side computer passes control signals and state data between the model predictive controller node and the Crazyflie ROS server. The Crazyflie server packages Tx PWM values to send and unpacks Rx compressed log data from the robot.

Our 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 2). 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 Hz to 250 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 2) 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.

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

IV. Learning Forward Dynamics

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 $s_{t+1} = s_t + f_\theta(s_t, a_t)$. 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 [3].

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 [25] with a learning rate of .0005 and a batch size of 32. The network design is summarized in Figure 3. All layers except for the output layer use the Swish activation function [26] 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.

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

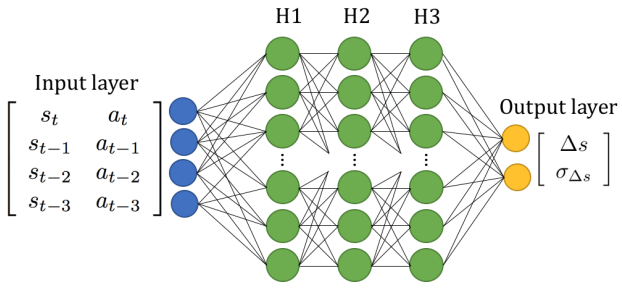

Fig. 3: The neural network dynamics model consists of the past 4 state-action pairs predicting the mean and variance of the change in state. The input is of dimension 52, predicting an 18 dimensional output. We use 3 hidden layers of width 500, with the Swish activation function. Training uses the Adam 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.

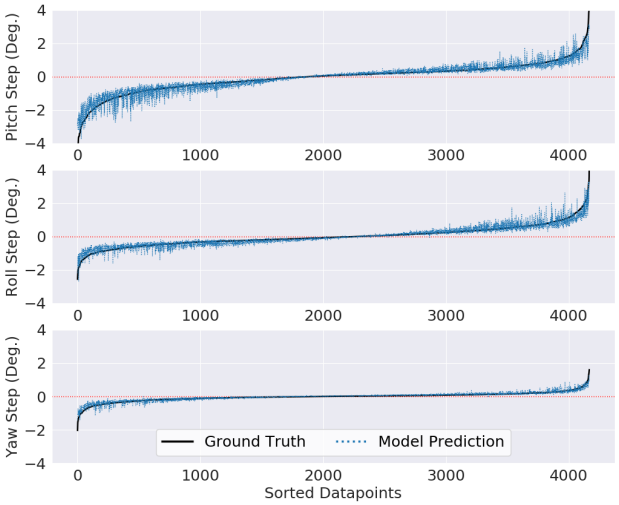

Fig. 4: One step predictions for change in Euler angles of the final model across the state space of data collected on the random action policy. The normalized mean squared errors from ground truth to the prediction for each of the Euler angles are Pitch: 0.130, Roll: 0.076, and Yaw: 0.032.

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. 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 4.

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 $250\,\text{ms}$, so our update time-step of $6.7\,\text{ms}$ 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 $25\,\text{ms}$ to reach the desired rotor speed; when operating at $150\,\text{Hz}$, the time step between updates is $6.7\,\text{ms}$, 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:

$$\xi_t = \begin{bmatrix} s_t & s_{t-1} & s_{t-2} & s_{t-3} & a_t & a_{t-1} & a_{t-2} & a_{t-3} \end{bmatrix}^T \tag{2}$$

V. Control Algorithm

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.

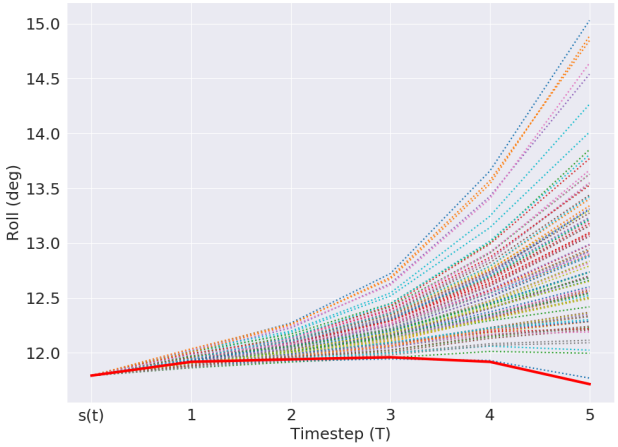

Fig. 5: Each line is a predicted future state from a candidate action, with the chosen "best action" highlighted in red. The near-linearity of the predictions at time-steps 3 to 5 indicates that the trade-off in control frequency is not worth the potential for increased predictive power of nonlinear effects. For example, at $N = 5000$, a time horizon of 2 will run at a control frequency of about $150\,\text{Hz}$, while a time horizon of 5 the frequency drops to $80\,\text{Hz}$.

The best action is decided by a user designed objective function that takes in the simulated trajectories $\hat{X}(a, s_t)$ and returns a best action, a^*; as visualized in Figure 5. 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.

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

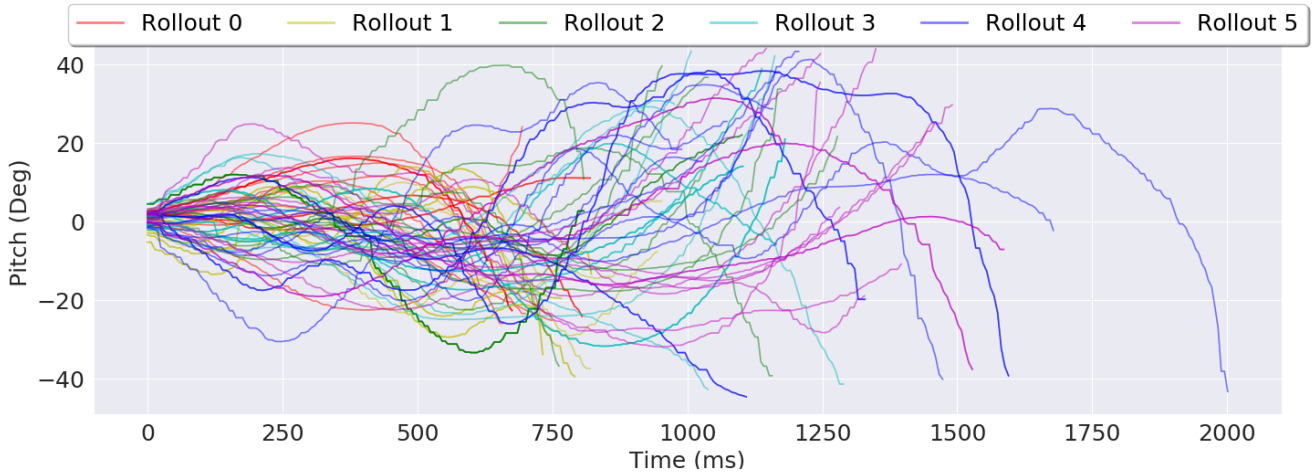

Fig. 6: Overlaid trajectories in pitch of the longest 15 flights from each model's roll-out. The data shows not only an increase in flight length for the controllers trained on more data, but also the ability to recover from more extreme states.

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

Our MPC operates on a short time horizon, $T = 2$ to leverage the predictive power of our model but maintain control frequency. The low predictive range of this controller makes the performance sensitive to damping factors in the objective function because dangerous actions will not strongly diverge on our prediction timescale. At $T = 1$, the controller can run over 200 Hz, but we do not have reliable derivative information on our predictions. Experiments show that for time horizons greater than $T = 2$ the neural network predictions are primarily linear, with areas of substantial jumps corresponding to untrained state-space. Linearity implies that the objective function will not gain substantial predictive power from a longer time horizon, while substantial prediction jumps would add instability to the chosen best action. On a system running with an Nvidia Titan Xp, we achieved a maximum control frequency of 230 Hz with $N = 5000, T = 1$. For testing we use a locked frequency of 150 Hz at $N = 5000$, $T = 2$. Our control algorithm is summarized in Algorithm 1.

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

VI. Evaluation

The performance of our controller is measured by the average flight length over each roll-out. 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.

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**
      $\{a\}_{i,j} \sim U\left(p_{eq,j} - \sigma, p_{eq,j} + \sigma\right)$
      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

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 5,209 points of dynamics data 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 open-loop takeoff followed by on-policy control while logging data to the ROS base-station. 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.

B. Performance Summary

The best results over the model roll-outs are shown in Figure 6, where later roll-outs are clearly longer on average. The data used for training and the results of the roll-outs is summarized in Table I, which shows the average flight

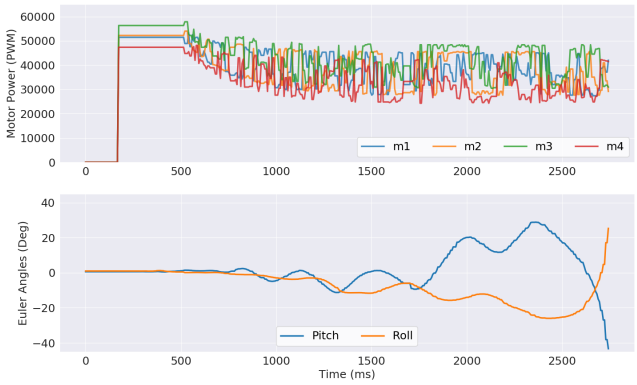

Fig. 7: The performance of the control algorithm over a test flight. The time under control for this flight is 2 s. Above: The open-loop takeoff followed by the controlled PWM values over time. Below: Controlled pitch and roll.

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 7, 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. Future iterations of a MPC with longer time horizons could mitigate this effect. 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/

R	Flights	Collected Points	Trained Points	Top 20 Flights Time (ms / flight)
0	91	5,209	0	686.6
1	173	8,616	5,091	843.7
2	301	15,480	13,348	980.1
3	165	12,098	28,559	1,207.1
4	188	14,446	39,861	1,332.3
5	222	16,320	55,731	1,401.1

TABLE I: The data statistics for each autonomous roll-out.

VII. Conclusions and Future Work

This work is an exploration of the capabilities of model-based reinforcement learning for the low-level control of an initially unknown system at high frequencies. 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 $150\,\mathrm{Hz}$. 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 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.

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 [27]–[29]. 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.

Acknowledgment

The authors thank the UC Berkeley Sensor & Actuator Center (BSAC), Berkeley DeepDrive, and Nvidia Inc.

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] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

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

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

[27] 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.

[28] 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.

[29] 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.