2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Deep Learning a Quadrotor Dynamic Model for Multi-Step Prediction

Nima Mohajerin, Melissa Mozifian and Steven Waslander[1]

Abstract— **We develop a multi-step motion prediction modeling method for dynamic systems over long horizons using deep learning. Building on previous work, we propose a novel hybrid network architecture, by combining deep recurrent neural networks with a quadrotor motion model created using classic system identification methods. The proposed model takes *only* the initial system state and motor speeds over the prediction horizon as inputs and returns robust state predictions for up to two seconds of motion at 100 Hz. We employ recurrent neural network state initialization during training, to exploit real-world dataset collected from quadrotor vehicle flights in an indoor flight arena. Our experiments demonstrate that the proposed hybrid network model consistently outperforms both black box and rigid body dynamics predictions over single and multi-step prediction scenarios, with an order of magnitude improvements in velocity estimates in particular.**

I. Introduction

Predicting the behaviour of a dynamic system multi-steps ahead in time has many applications [1], [2]. In a Model Predictive Control (MPC) scheme, for example, an accurate prediction of the plant's behaviour in the planning process can boost the control performance and allow the receding horizon to shift forward many steps at a time [3]. In a multi-step prediction scenario, the model is used in an open loop fashion; an input *sequence* (or *trajectory*) is given to the model along with an initial state and the model recursively predicts the system state over a prediction horizon without any additional information. This causes the unmodeled dynamics and accumulated noise error to drastically deteriorate the prediction accuracy as the prediction horizon increases.

Traditionally, a model is developed based on first principles and then the parameters of the model are identified using established system identification techniques [4], [5]. In this work, we refer to such a model as *white-box*, or a *first principles* model, as the derivation of the model is understood from first principles and therefore considered visible. Identifying the parameters of a white-box model may require extensive experimental procedures and must be repeated for every modification to the vehicle configuration. For instance, measuring the blade drag coefficient of a quadrotor requires a wind tunnel and altering the vehicle body can greatly influence the drag coefficients. Additionally, many properties of the system may be too difficult to accurately model from first principles. For instance, modeling the full suite of aerodynamic effects on quadrotor remains an open problem. Due to their light weight and large thrust to weight ratio, quadrotors are very susceptible to aerodynamic forces affecting their motion.

Modeling the dynamics of a quadrotor has been studied extensively in [6], [7], [8]. First principles models of quadrotors have also been used in control [8], [9], [10]. These models tend to rely on steady state aerodynamic models that do not capture rapid dynamic motions. As a result, they are primarily used in single-step predictions for vehicle control, where the prediction is required over one step into the future, given the current system state and the input. Using these models in a multi-step prediction fashion, the predicted states rapidly diverge as we will demonstrate in this work.

Few existing methods use deep learning techniques to model aerial vehicles. Abbeel et al. employed Feed Forward Neural Networks (FFNNs) with Rectified Linear Units (ReLUs) to model a real helicopter in a single-step fashion [11]. A similar approach was adopted in modeling a quadrotor [12], for single-step prediction and control. Recently black-box models (models that are fully learned from recorded data) have been developed using Recurrent Neural Networks (RNNs) for quadrotors [13], [14], [15], [16]. In [13], an Echo-State RNN is used to model a quadrotor from experimental data. Although not explicitly stated, their results show a single-step prediction scenario. In the author's previous work, a deep RNN is used to model the dynamic of the altitude of a real quadrotor. It is also shown that the model significantly outperforms the first principles model in multi-step prediction scenarios [16].

In this work, we apply our previously proposed approach in [16] to initialize states of deep Long-Short-Term-Memory (LSTM) networks [17] to generate a full black-box model of a real quadrotor. Furthermore, we develop a novel hybrid model for a quadrotor by combining the rigid-body dynamics with RNNs in two configurations. Our proposed hybrid model employs deep LSTM networks with state initialization in conjunction with a motion model of a quadrotor derived from laws of rigid-body dynamics. Our model learns the quadrotor model parameters and the network weights using the dataset we have collected from a real quadrotor. We evaluate an instance of the first principles model, the black-box and the hybrid model in multi-step prediction scenarios on the dataset and show that our hybrid model significantly outperforms both models. Our model predicts the vehicle velocity and body rates over almost 2 seconds with the errors less than 10 cm/sec and 7 deg/sec. In 40 steps prediction (0.4s flight time), the hybrid model errors are less than 4 cm/sec and 1.5 deg/sec.

[1]Nima Mohajerin, Melissa Mozifian and Steven Waslander are with Faculty of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON, Canada. (nima.mohajerin, melissa.mozifian, stevenw)@uwaterloo.ca

II. Multi-Step Prediction Problem Formulation

Consider a dynamic system $\mathcal{S}_n^m$ with m input and n output dimensions in the discrete time domain. The system input and output at a timestep, k, is denoted by $\mathbf{u}(k) \in \mathbb{R}^m$ and $\mathbf{y}(k) \in \mathbb{R}^n$, respectively. It is assumed that both input and output are measurable at all timesteps, k. Consider an input sequence of length T starting at a time instance $k_0 + 1$, $\mathbf{U}(k_0+1, T) \in \mathbb{R}^m \times \mathbb{R}^T$,

$$\mathbf{U}(k_0+1, T) = \begin{bmatrix}\mathbf{u}(k_0+1) & \mathbf{u}(k_0+2) & \dots & \mathbf{u}(k_0+T)\end{bmatrix}. \tag{1}$$

The system response to this input is an output sequence denoted by $\mathbf{Y}(k_0+1, T) \in \mathbb{R}^n \times \mathbb{R}^T$,

$$\mathbf{Y}(k_0+1, T) = \begin{bmatrix}\mathbf{y}(k_0+1) & \mathbf{y}(k_0+2) & \dots & \mathbf{y}(k_0+T)\end{bmatrix}.$$

Given an input sequence $\mathbf{U}(k_0+1, T)$, the multi-step prediction problem seeks an accurate estimate of the system output over the same time horizon, $\tilde{\mathbf{Y}}(k_0+1, T) \in \mathbb{R}^n \times \mathbb{R}^T$,

$$\tilde{\mathbf{Y}}(k_0+1, T) = \begin{bmatrix}\tilde{\mathbf{y}}(k_0+1) & \tilde{\mathbf{y}}(k_0+2) & \dots & \tilde{\mathbf{y}}(k_0+T)\end{bmatrix}, \tag{2}$$

which minimizes a measure of distance between the actual and predicted outputs. Typically, a Sum-of-Squares Error measure (SSE) is chosen,

$$L = \frac{1}{T} \sum_{k=k_0+1}^{k_0+T} \mathbf{e}(k)^\top \mathbf{e}(k), \tag{3}$$

$$\mathbf{e}(k) = \mathbf{y}(k) - \tilde{\mathbf{y}}(k). \tag{4}$$

In the above equations, $\mathbf{e}(k)$ is the prediction error at the kth prediction step.

III. Models for a Quadrotor

The quadrotor models described in this section are implemented in the discrete time domain. In a multi-step prediction scenario, they receive a sequence of motor speeds over a prediction length, as in (1), and generate a sequence of body rates and velocities, as in (2).

A. First Principles Model

We adopt the model described in [10]. For convenience we treat the variables in the continuous time domain in this section only. The quadrotor state vector is formed as follows,

$$\begin{aligned}\mathbf{x}^\top &= [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ x_{10}\ x_{11}\ x_{12}] \\ &= [\boldsymbol{\eta}^\top\ \boldsymbol{\omega}^\top\ \boldsymbol{\xi}^\top\ \dot{\boldsymbol{\xi}}^\top] \\ &= [\phi\ \theta\ \psi\ p\ q\ r\ x_I\ y_I\ z_I\ \dot{x}_I\ \dot{y}_I\ \dot{z}_I],\end{aligned} \tag{5}$$

where ϕ, θ and ψ are the Euler angles, i.e. roll, pitch and yaw, respectively (Figure 1). Body rates are denoted by $\boldsymbol{\omega}^\top = [p, q, r]$ and position by $\boldsymbol{\xi}^\top = [x_I, y_I, z_I]$. The equations governing the dynamics of a quadrotor are,

$$\begin{aligned}\dot{x}_1 &= x_4 + x_5 S_1 T_2 + x_6 C_1 T_2 \\ \dot{x}_2 &= x_5 C_1 - x_6 S_1 \\ \dot{x}_3 &= \sec(x_2)(x_5 S_1 + x_6 C_1) \\ \dot{x}_4 &= -((I_z - I_y)/I_x) x_5 x_6 - (k_r x_4 / I_x) + (1/I_x)\tau_p \\ \dot{x}_5 &= -((I_x - I_z)/I_y) x_4 x_6 - (k_r x_5 / I_y) + (1/I_y)\tau_q \\ \dot{x}_6 &= -((I_y - I_x)/I_z) x_4 x_5 - (k_r x_6 / I_z) + (1/I_z)\tau_r \\ \dot{x}_7 &= x_{10}, \quad \dot{x}_8 = x_{11}, \quad \dot{x}_9 = x_{12} \\ \dot{x}_{10} &= (-k_t x_{10}/m) + (1/m)(C_1 S_2 C_3 + S_1 S_3)\tau_f \\ \dot{x}_{11} &= (-k_t x_{11}/m) + (1/m)(C_1 S_2 S_3 - S_1 C_3)\tau_f \\ \dot{x}_{12} &= (-k_t x_{12}/m) + (1/m)(C_1 C_2)\tau_f - g,\end{aligned} \tag{6}$$

where, I_x, I_y, I_z represent inertia, m represents mass of the quadrotor, k_t and k_r represent the translational and rotational drag coefficients and are assumed to be constant in all directions for simplicity. Also, $T_i = \tan(x_i)$, $S_i = \sin(x_i)$ and $C_i = \cos(x_i)$. The control inputs are represented by τ_p, τ_q, τ_r, which are the moments about the three axes, and τ_f, which is the total thrust acting on the vehicle. The acceleration due to gravity is represented by g.

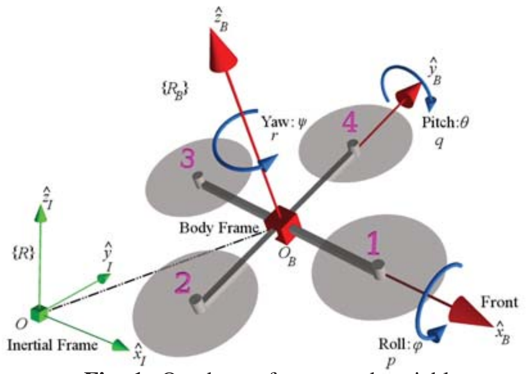


Fig. 1: Quadrotor frames and variables.

In [10], the control inputs are generated by the following mapping from the motor speed,

$$\begin{bmatrix}\tau_f \\ \tau_p \\ \tau_q \\ \tau_r\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 1 \\ 0 & -l & 0 & l \\ -l & 0 & l & 0 \\ -d & d & -d & d\end{bmatrix} \begin{bmatrix}f_1 \\ f_2 \\ f_3 \\ f_4\end{bmatrix}, \tag{7}$$

given the directions and motor numbering depicted in Figure 1. In equation (7) l is the distance from the center of mass to the rotors, d is the ratio between the drag and the thrust coefficients of the blade, and f_i for $i = \{1, 2, 3, 4\}$ are the forces generated by the four rotors of the quadrotor which are related to the rotor speeds, u_i, by the following equation,

$$f_i = \frac{b_i + u_i^2}{k_i} \tag{8}$$

where b_i and k_i are motor constants and are identified for each motor. The mapping depicted by equation (7) is only used for the white-box model in this work.

B. Full Black-box Model

Using RNNs, it is possible to learn a dynamic model of a quadrotor from observations [14], [15]. In a multi-step prediction scenario, the input to the RNN is a sequence of the motor speeds and the RNN generates the quadrotor states. Therefore, each training instance is a tuple of two equal-length time-series, an input, which is the motor speeds, and an output, which includes the quadrotor states. As described in [16], an RNN should be properly initialized to accurately predict the system states. The authors in [16] addressed the state initialization problem and demonstrated that the proposed architecture for modeling the altitude of a real quadrotor outperforms the first principles model described above. This work applies the same initialization method to LSTM networks.

The state initialization procedure proposed in [16] splits the input and output sequences of each training instance into two segments. The first segment of the input and output are fed to an FFNN (the initializer network) which generates the initial state values for the RNN (the predictor network). The second segment is used for prediction; the input segment is fed to the RNN and it is trained to learn the output segment. In this work, we adopt the same method to initialize the states of LSTM networks to learn a full black-box model. The gated architecture of LSTM networks helps to retain relevant information across time. There are many versions of LSTMs [18]. In this work, we use the version described in [19].

In this work, we use networks with multiple layers of LSTM cells connected in a serial fashion. To feed the networks a complete history over a truncated horizon, the networks can also be equipped with input and output buffers or *Tapped Delay Lines* (TDLs) [20]. The following notation is used to state a network architecture, network-type : number of layers $\times$ number of cells. For instance, a network with 5 layers, each having 200 cells, and input/output TDLs is depicted as LSTM TDL:5$\times$200.

C. Hybrid Model

We propose a hybrid model which comprises three modules; an Input Model (IM), a Motion Model (MM) and an Output Model (OM). Using rigid body dynamics, the MM module implements the motion model of a quadrotor i.e. the descritized version of the equations (6). The IM module provides the input to the MM module, $\tilde{\boldsymbol{\tau}}$, which has four elements that are plugged into the equations (6) as τ_p, τ_q, τ_r and τ_f. The OM module compensates for the deviation of the MM module output from the desired values. The IM and OM are deep RNNs with LSTM cells initialized by the initialization scheme discussed in [16]. Depending on the three modules connections, two architectures are proposed:

*1) **Hybrid-Serial Configuration**:* This configuration is illustrated in Figure 2. Blue links represent feedback connections and black links represent feedforward connections. The model output is directly provided by the OM module. The gains $\mathbf{w}_\omega$ and $\mathbf{w}_{\dot{\xi}}$ are normalization factors. Note that the output of the MM does not receive any penalty directly; the error is assessed on the OM module only, and reaches the MM and IM modules through backpropagation. The hybrid-serial configuration assumes no prior relation between part of the system that is modelled by the MM module and the unmodelled part, which is learned from the observations during training.

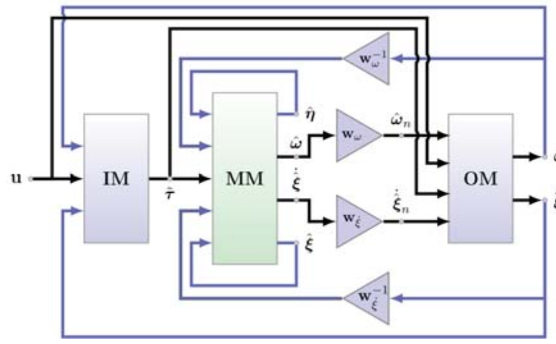


Fig. 2: Proposed hybrid-serial model.

*2) **Hybrid-Parallel Configuration**:* A possible scenario in the hybrid-serial configuration is that the RNNs take control and attenuate the influence of MM. Since RNNs have many degrees of freedom, this case is not necessarily inevitable. One possible solution is to directly incorporate the output of the MM module in the model output. This is depicted in Figure 3. The underlying assumption in this configuration is that the modeling error of the MM module can be represented by an additive term, i.e. the output of the OM module. In this configuration, the OM module acts as a compensator which predicts and eliminates error in the MM module.

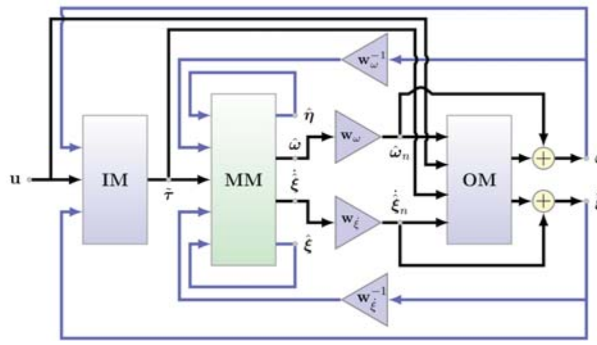


Fig. 3: Proposed hybrid-parallel model.

IV. The Quadrotor Dataset

The modeling approaches considered in this work are applied to the AscTec Pelican, a light-weight quadrotor equipped with a real-time autopilot board coupled with an onboard computer (AscTec Mastermind). The onboard computer communicates with the autopilot board via a UART connection and runs a ROS node[1] to record motor speeds and Inertial Measurement Unit (IMU) readings. The vehicle is operated in an indoor environment by an expert pilot using a Futaba T7C remote control. The dataset we have collected is publicly available.[2]

[1]http://wiki.ros.org/asctec_mav_framework
[2]https://github.com/wavelab/pelican_dataset

The vehicle position and inertial orientation are measured at 100 Hz using a Vicon motion capture system, equipped with 16 Vantage cameras looking at the IR reflective markers mounted on the vehicle. These measurements are then sent to the Vicon server through a LAN communication. To avoid any wireless latency and/or packet drops, the measurements are logged on the Vicon server computer using the Vicon Tracker software version 3.3 and then synchronized with the autopilot measurements offline. The Vicon system is calibrated before each data collection session to account for changes in environmental variables, such as room temperature, camera body temperature, etc.

The quantities included in the quadrotor dataset are

- motor speed, $\mathbf{u}(k) = [u_1(k), u_2(k), u_3(k), u_4(k)]$,
- velocity vector, $\dot{\boldsymbol{\xi}}(k) = [\dot{x}_I(k), \dot{y}_I(k), \dot{z}_I(k)]$,
- body rates, $\dot{\boldsymbol{\omega}}(k) = [p(k), q(k), r(k)]$.

The dataset consists of various flight regimes; hover, close to ground, light, moderate and aggressive manoeuvres in all directions. The dataset total flight time is approximately 3 hours and 50 minutes. The maximum attained values for the vehicle are listed in Table I. There are approximately 1.39 million samples per each element in our dataset. The dataset is split 60% for training and 40% for generalization testing.

$\dot{x}\ (m/s)$	$\dot{y}\ (m/s)$	$\dot{z}\ (m/s)$	$p\ (rad/s)$	$q\ (rad/s)$	$r\ (rad/s)$
3.9268	3.9721	5.8526	3.9116	3.8506	3.7902

TABLE I: Maximum values for the Pelican measurements.

V. Results and Discussions

The quadrotor is modelled as a Multi-Input-Multi-Output (MIMO) system, which receives the motor speeds trajectory and predicts the vehicle body rates and velocity (in inertial frame). The black-box and hybrid models are implemented using Google's Tensorflow library [21] in Python. The network training and inference was performed using a GeForce Titan Xp GPU. For evaluation, the prediction error and its distribution are studied. The prediction error is defined at each prediction step as in equation (4). We take the L1 norm of the velocity and the body rate error vectors and compare the distribution over the prediction steps,

$$\begin{aligned} \bar{e}_{\dot{\xi}}(k) =& \tfrac{1}{3}\big(|e_{\dot{x}_I}(k)| + |e_{\dot{y}_I}(k)| + |e_{\dot{z}_I}(k)|\big), \\ \bar{e}_{\omega}(k) =& \tfrac{1}{3}\big(|e_p(k)| + |e_q(k)| + |e_r(k)|\big), \end{aligned} \tag{9}$$

where $\bar{e}_{\dot{\xi}}(k)$ is in meters per second (m/s) and $\bar{e}_{\omega}(k)$ is in degrees per second (deg/sec). Using absolute values weighs all errors equally, regardless of sign. Using L2 norm does not significantly alter the comparative results.

A. White-box parameter identification

The drag coefficients and the motor parameters for the first principles model are identified using a Least-Squares (LS) method. Since the first principles model is valid around hover, LS is applied to the near hover flight regimes. The inertias (I_x, I_y and I_z), mass m and the identified parameters are listed in Table II. Note that the flight volume is not large

k_r	k_t	b_1	k_1	b_2
0.0099	2.35×10^{-14}	0.062	7.63×10^{6}	0.082

k_2	b_3	k_3	b_4	k_4
1.21×10^{7}	0.0016	4.67×10^{6}	2.28×10^{-4}	4.69×10^{6}

$I_x(kg.m^2)$	$I_y(kg.m^2)$	$I_z(kg.m^2)$	$m(kg)$	$l(m)$
0.002	0.002	0.001	1.6	0.211

TABLE II: Quadrotor white-box parameters.

enough to capture the drag coefficient, k_r. As a result LS identifies this parameter close to zero.

For validating the model, however, the entire dataset is used. The model is employed in a single-step and a 20-step prediction scenario.

B. Black-box model

Initially, the black-box model was trained on the MIMO system. However, the velocity predictions diverged rapidly during training preventing the learning to converge. Therefore, the velocity and body rates were decoupled as two MIMO sub-systems to be learned by two separate RNNs. The first RNN is trained to predict the body rates from the motor speeds. The second RNN uses the predicted body rates along with the motor speeds to predict the velocity. However, we train the second RNN on the *actual* body rates. This form of training resembles teacher forcing [22]. Note that in a multi-step prediction scenario, the second RNN must use the *predicted* body rates.

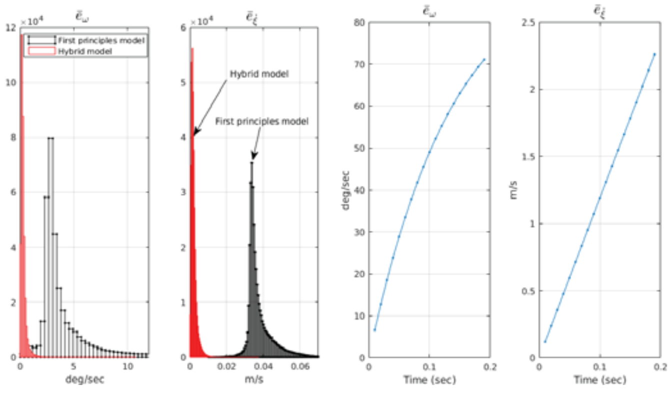
C. Hybrid model

In contrast to the black-box model, the hybrid model learns the full MIMO system. The model receives the motor speeds and generates prediction for the velocity and body rates *at the same time*. The drag coefficient parameters of the MM model is identified as part of the learning process. The identified values from the hybrid-serial model were not as consistent as the values form the hybrid-parallel model, implying the possibility that the MM module was not being effectively used. The identified parameters by the hybrid-parallel model were $k_r = 0.026 \pm 0.0132$ and $k_t = (2.1 \pm 1.6) \times 10^{-8}$.

D. Comparisons

In Figure 4, the white-box performance is studied. In Figure 4a the single-step prediction error distributions are plotted for the white-box and the hybrid-parallel models and the performance boost of the hybrid model is shown. In Figure 4b, the white-box is used in a multi-step prediction scenario for 20 steps. It is observed that the prediction error rapidly diverges. Comparisons to the proposed RNN multi-step predictors are, therefore not included, as they differ by too great an extent to be properly visualized.

Figure 5 compares the prediction performance of the black-box model, the hybrid-serial and the hybrid-parallel configurations for $T = 50$ timesteps. For each model, LSTM networks are used with and without TDLs. For the TDL, the buffer size is 10. Our experiments show that incorporating TDL on average slightly improves the black-box and the hybrid-serial configurations at the cost of slight increase in the network size. However, for the parallel configuration the



(a) Single-step prediction **(b)** Multi-step prediction

Fig. 4: White-box prediction performance. **Left**: histograms of the velocity and body rate prediction errors for the white-box and hybrid-parallel model. **Right**: white-box error increase over 20 steps in multi-step prediction scenario.

TDL causes slight decrease in the performance. Note that we have used the actual body rates for the black-box velocity prediction. Using state measurements over the prediction horizon contradicts the definition of the multi-step prediction problem. Nevertheless, the hybrid model outperforms the black-box model consistently, despite the black box model's use of real body rate measurements. From this point forward in the results, we will use the predicted body rates to generate velocity prediction for the black-box model, significantly decreasing black-box performance.

This dominance of the hybrid parallel method held over four distinct prediction lengths ($T = 50, 100, 150$ and 200 steps, or 0.5, 1.0, 1.5 and 2.0 seconds flight time, respectively) with similar trends in error growth for all experiments. In the interests of space saving, we restrict detailed presentation of multi-step prediction results to a comparison of the black-box model with predicted body rates and the hybrid-parallel model, over horizons of 50 and 200 steps. Note that the first 10 steps are used for the initialization procedure, hence, the plot lengths are $T - 10$ steps.

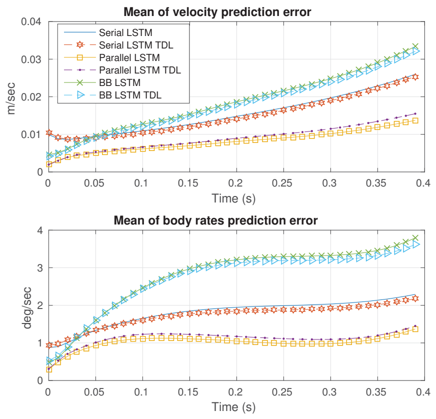


Fig. 5: Comparison between the black-box and the hybrid models ($T = 50$). The black-box (BB) architectures are LSTM:7×200 and LSTM TDL:7×200. The IM and OM modules in hybrid models are both either LSTM:4×200 or LSTM TDL:4×200.

In Figures 6, 7, 8 and 9 we compare the error distributions of the black-box model and the hybrid-parallel model for the two prediction lengths. The black-box and the hybrid-parallel architecture are LSTM TDL:7×200 and LSTM:4×200 (both IM and OM), respectively. For the black-box model, we use the predicted body rates to predict velocity. For the shortest prediction length ($0.4s$) the hybrid-parallel model reduces the body rate error by 50% and the velocity error by about $20\times$ compared to the black-box model. The reason that the black-box model performs poorly on the velocity prediction, as stated before, is that we use the predicted body rates.

For $T = 200$ and body rate prediction, the black-box initially performs worse, however, in the long run it performs better than the hybrid model. One possible explanation for this behaviour is that the MM module in the hybrid model limits the exploration capacity of the OM module. Although, the inclusion of the motion model contributes significantly to the accuracy of the early prediction steps, it may also impose restrictions on the search of optimal weights for the RNNs. However, the LSTMs in the black-box model are free to explore the entire weight space and they can accumulate relevant information over longer time to perform better. For the velocity, we see that black-box still performs worse, however, a slight decrease is seen on the later predictions which is similar to the behaviour in the body rate prediction.

In Figure 10, the mean of the prediction errors are illustrated before (the MM output) and after compensation for each prediction length. This results show the importance of RNN network initialization. The plots show that the MM module prediction error starts from a non-zero value which requires the OM module to immediately compensate for that. Therefore, the output of the OM module should start from a non-zero value which requires a proper state initialization for the RNNs. Note that at each prediction step, the compensated states are fed into the MM module.

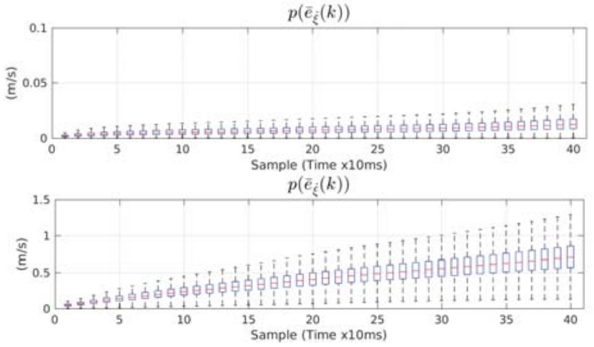


Fig. 6: Velocity error distribution of the hybrid-parallel (top) and the black-box (bottom) models ($T = 50$). Note that hybrid-parallel prediction is over an order of magnitude better than the black-box.

VI. Conclusion

We propose a hybrid model for a real quadrotor in a multi-step prediction scenario. Our model takes advantage of prior knowledge imposed by the first principles model and in conjunction with deep RNNs, it learns the underlying quadrotor dynamics from a real quadrotor dataset. Our approach significantly outperforms existing models based on first principles modelling as well as black box modelling. Our predictions show competitive performance with the state-of-the art quadrotor modelling techniques.

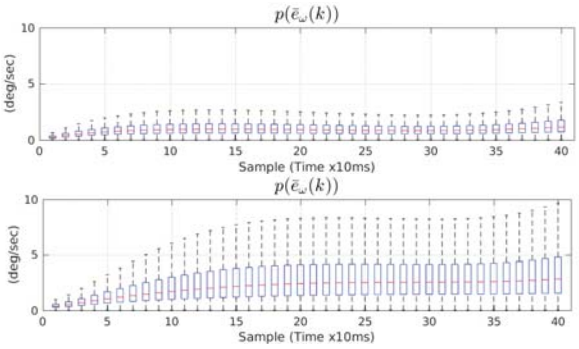


Fig. 7: Comparison of the body rate error distribution between the hybrid-parallel (top) and the black-box (bottom) models ($T = 50$).

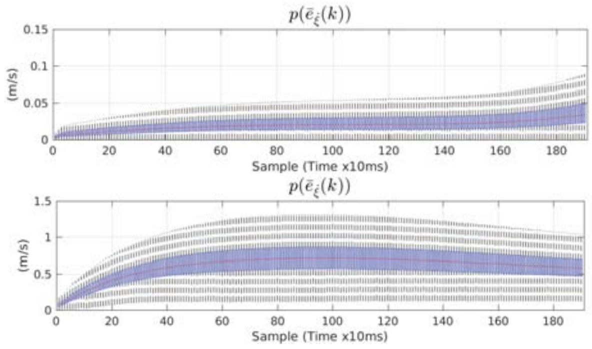


Fig. 8: Comparison of the velocity error distribution between the hybrid-parallel (top) and the black-box (bottom) models ($T = 200$).

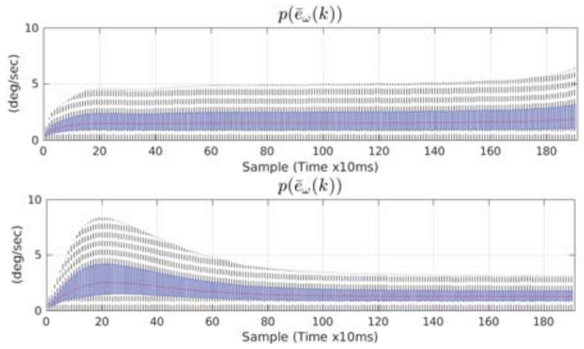


Fig. 9: Comparison of the body rate error distribution between the hybrid-parallel (top) and the black-box (bottom) models ($T = 200$).

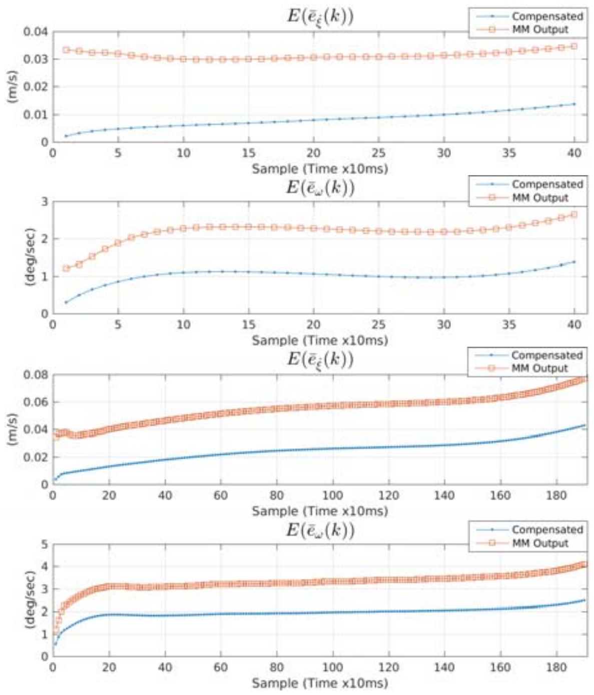


Fig. 10: The means of uncompensated output of the MM module vs. the compensated output for the velocity and body rate predictions.

References

[1] A. G. Parlos, O. T. Rais, and A. F. Atiya, "Multi-step-ahead prediction using dynamic recurrent neural networks," *Neural networks*, vol. 13, no. 7, pp. 765–786, 2000.

[2] R. Boné and M. Crucianu, "Multi-step-ahead prediction with neural networks: a review," *9emes rencontres internationales: Approches Connexionnistes en Sciences*, vol. 2, pp. 97–106, 2002.

[3] J. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[4] L. Ljung and T. Söderström, *System identification*. MIT Press, 1983.

[5] T. P. Bohlin, *Practical grey-box process identification: theory and applications*. Springer Science & Business Media, 2006.

[6] T. Madani and A. Benallegue, "Adaptive control via backstepping technique and neural networks of a quadrotor helicopter," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 6513–6518, 2008.

[7] 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.

[8] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Precision flight control for a multi-vehicle quadrotor helicopter testbed," *Control engineering practice*, vol. 19, no. 9, pp. 1023–1036, 2011.

[9] H. Voos, "Nonlinear control of a quadrotor micro-uav using feedback-linearization," in *Mechatronics, 2009. ICM 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[10] A. Freddi, A. Lanzon, and S. Longhi, "A feedback linearization approach to fault tolerance in quadrotor vehicles," in *Proceedings of The 2011 IFAC World Congress, Milan, Italy*, 2011.

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

[12] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 4653–4660.

[13] A. Vargas, M. Ireland, and D. Anderson, "System identification of multi-rotor uavs using echo state networks," 2015.

[14] N. Mohajerin and S. L. Waslander, "Modular deep recurrent neural network: Application to quadrotors," in *2014 IEEE International Conference on Systems, Man and Cybernetics*, 2014.

[15] ——, "Modelling a quadrotor vehicle using a modular deep recurrent neural network," in *2015 IEEE International Conference on Systems, Man and Cybernetics*, 2015.

[16] ——, "State initialization for recurrent neural network modeling of time-series data," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 2330–2337.

[17] J. Schmidhuber and S. Hochreiter, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE transactions on neural networks and learning systems*, 2017.

[19] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[20] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 1999.

[21] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[22] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4 –27, March 1990.