Estimating Wind Velocity using a Neural Network with Quadcopter Trajectories

Sam Allison*, He Bai† , Balaji Jayaraman‡
Oklahoma State University, Stillwater, Oklahoma, 74078

In this paper, we examine the viability of using long short-term memory neural networks as a tool to estimate wind velocity from quadcopter trajectory information, such as position and Euler angles. We train a neural network on the deviations that a quadcopter experiences when subjected to wind disturbances and use the trained neural network to estimate wind velocities given a new trajectory. We conduct simulations to evaluate the estimation errors for constant winds and winds generated using the Dryden model. The simulation results demonstrate encouraging wind estimation performance using a quadcopter's trajectory and pitch and roll angles.

I. Introduction

Wind velocity measurements are crucial in many fields, such as aerospace, meteorology, and environmental sciences. For aerospace applications, these measurements are necessary in order to determine bounds on potential trajectory deviations, fuel usage, and whether or not conditions are safe for flight. Small UAS in particular are sensitive to wind gusts, and for safe flight near buildings, people, or other aircraft, the wind velocity must be accounted for when planning flight paths. Since small UAS have recently seen widespread adoption in a number of fields such as search and rescue, filming of TV shows and movies, and surveying land and crops, wind velocity measurements have become increasingly important. However, obtaining accurate wind velocity measurements over large areas is difficult to accomplish with the current approaches, such as ground stations, weather balloons, etc.

A wind sensor may be mounted on a quadcopter for wind measurements. However, it reduces the payload capacity of a quadcopter and can be expensive compared to the price of the quadcopter. In this paper, we investigate measuring wind using quadcopter trajectory information without a wind sensor. Since quadcopters are highly mobile and are sensitive to disturbances such as wind, leveraging trajectory deviation data from flights may provide wind velocity information that could be used to build a higher fidelity wind map for the flight area. However, since many of these commercially available quadcopters have proprietary controllers, it is difficult to predict the disturbance response of a given quadcopter. Our goal is to find an effective technique to model the relationship between wind velocity and quadcopter trajectories without knowledge of the onboard controller so that we can estimate wind velocity data directly from trajectory data.

In our previous work [1], we consider using a linear difference equation approach to modeling the trajectory deviations of a quadcopter subject to crosswind disturbances. Coefficients of a difference equation are identified based on training data using least squares. In this paper, we consider the horizontal 2-dimensional wind. Because of the nonlinear coupling in the drag term, we pursue a nonlinear modeling approach to estimate wind velocity using quadcopter trajectories. In particular, neural networks (NNs) have been widely used to model complex nonlinear dynamic systems [2, 3]. Recurrent NNs (RNNs) are commonly used for sequential and dynamic system modeling due to their ability to incorporate data from previous time steps into their predictions [4, 5]. Therefore, we have focused on using Long Short-Term Memory (LSTM) RNNs to generate the results for this paper.

While there are quite a number of publications on the topic of wind estimation using small UAS, the majority of them require either detailed knowledge of the dynamics and controller [6, 7], wind sensors [8, 9], constant airspeed, or hovering assumptions [10, 11]. We seek to provide a trajectory based approach that does not rely on knowledge of the controller or require extra instrumentation and does not assume a steady state velocity. A problem similar to this wind estimation problem is the trajectory deviation prediction of a quadcopter in a known wind field. A paper on trajectory

*Graduate Student, Student Member of AIAA, School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK, 74078
†Assistant Professor, School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK 74078
‡Assistant Professor, Senior member of AIAA, School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK 74078

deviation prediction that uses a similar strategy as our wind estimation approach is Xue's [12], where the author trains a NN on quadcopter position and velocity deviations and wind speeds and uses the NN to predict the quadcopter's trajectory when subjected to a new crosswind.

The remainder of this paper is structured as follows. Section II introduces the quadcopter dynamics and the controller used in simulations. Section III details the LSTM NN. In Section IV, the specific parameters and training data of the NN used to generate the results are presented. Section V provides results generated using the NN. Section VI discusses the conclusions and future work.

II. Quadcopter Modeling

In this section, we present the dynamic model of a quadcopter and a controller used in our study. The parameters of the model are based on the 3DR Iris+ [13] and the brushless DC motor model is based on a different BLDC motor [14] since the specific parameters for the Iris+ motors were not available. We do not include potential sensor errors in our model.

A. Dynamics

We use a standard formulation for the quadcopter dynamics [15], with the addition of a nonlinear drag term,

$$\begin{bmatrix} \ddot{p}_n \\ \ddot{p}_e \\ \ddot{p}_d \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} (-\cos\phi\sin\theta\cos\psi - \sin\phi\sin\psi)\frac{F}{m} + \frac{f_{d,n}}{m} \\ (-\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi)\frac{F}{m} + \frac{f_{d,e}}{m} \\ g - (\cos\phi\cos\theta)\frac{F}{m} + \frac{f_{d,d}}{m} \\ \frac{J_y - J_z}{J_x}\dot{\theta}\dot{\psi} + \frac{1}{J_x}\tau_\phi \\ \frac{J_z - J_x}{J_y}\dot{\phi}\dot{\psi} + \frac{1}{J_y}\tau_\theta \\ \frac{J_x - J_y}{J_z}\dot{\phi}\dot{\theta} + \frac{1}{J_z}\tau_\psi \end{bmatrix}, \tag{1}$$

where (p_n, p_e, p_d) are the north, east, and down positions in the inertial frame, (ϕ, θ, ψ) are the roll, pitch, and yaw, $(F, \tau_\phi, \tau_\theta, \tau_\psi)$ are the force and the moments in the designated directions, and $(f_{d,n}, f_{d,e}, f_{d,d})$ are the drag forces in the specified directions. Since the drag is a function of the total airspeed of the quadcopter, i.e., the difference between the wind velocity and the groundspeed of the quadcopter, we adapt the standard one-dimensional drag equation to

$$f_d = C_d(V_w - \dot{p})|V_w - \dot{p}|, \tag{2}$$

where V_w is the wind velocity vector in the inertial frame and C_d is the drag coefficient matrix. The matrix C_d was determined by fitting an exponential to the calculated drag coefficient at different airspeeds of the 3DR Iris+ model in Gazebo [13], resulting in $C_d = diag(min(1.1, (0.2 + 0.9\exp(-0.6|V_w - \dot{p}| - 2))))$.

The motor model used in Simulink is a third order transfer function given by

$$H(s) = \frac{2057342}{s^3 + 189.5s^2 + 13412s + 142834}, \tag{3}$$

where the input is pulse width modulation (PWM) and the output is the rotor angular rate. A PID motor controller is implemented and tuned to ensure that the motor reaches a desired angular rate within 0.2 seconds of receiving a desired value. To convert the rotor speeds to force and torques on the quadcopter, we use the following mapping

$$\begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k_1 & k_1 & k_1 & k_1 \\ 0 & -Lk_1 & 0 & Lk_1 \\ Lk_1 & 0 & -Lk_1 & 0 \\ -k_2 & k_2 & -k_2 & k_2 \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \tag{4}$$

where k_1 is the thrust coefficient, L is the quadrotor arm length, and k_2 is the motor torque coefficient.

B. Rotor Aerodynamics

In addition to the quadcopter and motor dynamics given above, the rotors of the quadcopter have a number of aerodynamic effects that can impact the performance of the quadcopter. Our model incorporates two of the more significant aerodynamics effects: air-relative velocity and blade flapping [16]. Air-relative velocity effects are due to the flow of the air through the rotor. In the zero-airspeed case, the surrounding air flows through the rotors at the expected rate, resulting in a thrust calculated as

$$T = k_1 \omega^2, \tag{5}$$

where T is the thrust of the rotor and ω is the angular rate of the rotor. However, if there is a non-zero airspeed, the thrust of each rotor needs to be corrected to account for the difference in the air flow. This corrected thrust can be calculated for each rotor as

$$T_{corrected} = \frac{T v_i}{v_i + w}, \tag{6}$$

$$v_i = \frac{v_h^2}{\sqrt{u^2 + v^2 + (v_i + w)^2}}, \tag{7}$$

where v_h is the induced velocity of the rotor while the quadcopter is hovering and (u, v, w) are the air velocities in the body frame.

Blade flapping is a 'tilting' of the rotor due to unequal forces on the advancing and retreating edges of the blade while in flight. The advancing edge of the blade has a higher airspeed than the retreating edge of the blade. Since the higher airspeed results in a higher thrust on the advancing edge, the rotor bends slightly more on the advancing edge than the retreating edge, resulting in a shift in the thrust plane to be away from the body frame. This necessitates defining the thrust from each rotor as a vector in the body frame, given by

$$T_{flapping} = \begin{bmatrix} \frac{u}{\sqrt{u^2+v^2}} \sin\alpha \\ \frac{v}{\sqrt{u^2+v^2}} \sin\alpha \\ \cos\alpha \end{bmatrix} T, \tag{8}$$

$$\alpha = K_f \sqrt{u^2 + v^2}, \tag{9}$$

where α is the blade flapping angle and K_f is the flapping coefficient.

C. Controller

To control the quadcopter, we use a feedback linearized PD attitude controller and a saturated PID waypoint navigation controller. Our saturated PID controller is given by

$$\begin{bmatrix} \phi^d \\ \theta^d \\ \psi^d \\ \ddot{p}_d^d \end{bmatrix} = \begin{bmatrix} \text{sat}(k_p e_{p_e} + k_d \dot{e}_{p_e} + k_i \int e_{p_e}, \phi\text{max}) \\ \text{sat}(k_p e_{p_n} + k_d \dot{e}_{p_n} + k_i \int e_{p_n}, \theta\text{max}) \\ \psi \\ k_p e_{p_d} + k_d \dot{e}_{p_d} + k_i \int e_{p_d} \end{bmatrix}, \tag{10}$$

where the d superscript designates a desired value, k_p, k_i, and k_d represent the proportional, integral, and derivative control gains, ϕmax and θmax are the maximum allowed roll and pitch angles, and $e_p = p^d - p$ are the position errors in the designated directions. We use the saturation function

$$\text{sat}(x, x\text{max}) = \begin{cases} -x\text{max} & \text{if } x < -x\text{max} \\ x & \text{if } -x\text{max} \le x \le x\text{max} \\ x\text{max} & \text{if } x > x\text{max} \end{cases}, \tag{11}$$

to ensure that the waypoint navigation controller does not request an attitude angle that the quadcopter cannot achieve The desired Euler angles and vertical acceleration generated by (10) are then used in the attitude controller below to

generate desired force and moment commands

$$\begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \frac{m(g+\ddot{p}_d^d)}{cos(\phi)cos(\theta)} \\ J_x(-K_1\dot{\phi} - \frac{J_y-J_z}{J_x}\dot{\theta}\dot{\psi}) + K_{p1}e_1 + K_{d1}\dot{e}_1 \\ J_y(-K_2\dot{\theta} - \frac{J_z-J_x}{J_y}\dot{\phi}\dot{\psi}) + K_{p2}e_2 + K_{d2}\dot{e}_2 \\ J_z(-K_3\dot{\psi} - \frac{J_x-J_y}{J_z}\dot{\phi}\dot{\theta}) + K_{p3}e_3 + K_{d3}\dot{e}_3 \end{bmatrix}, \tag{12}$$

where

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \phi^d - \phi \\ \theta^d - \theta \\ \psi^d - \psi \end{bmatrix}. \tag{13}$$

Using this controller allows us to set desired positions for the quadcopter to reach while applying wind disturbances to the system.

D. Wind Estimation Problem

We investigate the problem of estimating V_w based on the state information of the quadcopter, such as (p_n, p_e, p_d) and (ϕ, θ, ψ), without knowledge of the controller used on the quadcopter. Since dynamic systems are most commonly approximated with RNNs, we propose to use a common form – long short-term memory (LSTM) NNs – to model the relationship from $(p_n, p_e, p_d, \phi, \theta, \psi)$ to V_w. We next provide a brief description of LSTM NNs.

III. The LSTM Neural Network Model

LSTM NNs are frequently used for dynamic system modeling due to their ability to learn information about long term dependencies in data [17]. One of the distinguishing features of LSTM NNs is their gated self-loops. Since the weighting on the gates is trained, it allows for the retention of information over long sequences of data. This mitigates the exploding/vanishing gradient problems experienced with some other forms of RNN.

LSTM NNs make predictions based on sequences of data. Each set of input sequences, X, with n timesteps of the input is paired with a single timestep of the target vector, T. The NN is trained on X and T and, after training, can be used to generate predictions for T when given a new X.

The structure of a LSTM unit is shown in Figure 1. LSTM units are composed of three gates that serve different purposes, as well as a memory cell to store previous inputs. Each gate has a sigmoid activation function, resulting in a vector with values between zero and one. Training with this activation function allows the gate to prioritize the data most relevant to the target values. The input gate given by

$$i_i^k = \sigma\left(b_i^i + \sum_j U_{i,j}^i x_j^k + \sum_j W_{i,j}^i h_j^{k-1}\right) \tag{14}$$

acts on a combination of the newest input data point and n previous data points (stored in the memory cell), where b_i^i is the input gate's bias for the i^{th} unit of the NN, U_i^i are the input weights, W_i^i are the recurrent weights, x^k are the inputs to the LSTM unit, and h^{k-1} are the internal states of the LSTM unit from the previous time step for j input features. In order to determine which of the previous inputs remains stored in the cell, the forget gate output given by

$$f_i^k = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^k + \sum_j W_{i,j}^f h_j^{k-1}\right) \tag{15}$$

is multiplied by the data in the memory cell at each time step. Thus, the internal state of the LSTM cell is given by

$$c_i^k = f_i^k c_i^{k-1} + i_i^k \sigma\left(b_i + \sum_j U_{i,j} x_j^k + \sum_j W_{i,j} h_j^{k-1}\right). \tag{16}$$

Once the new input and previous inputs have been appropriately scaled by the input and forget gates, they are sent through a hyperbolic tangent function as

$$h_i^k = \tanh(c_i^k) o_i^k, \tag{17}$$

and multiplied by the output of the output gate as

$$o_i^k = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^k + \sum_j W_{i,j}^o h_j^{k-1}\right), \tag{18}$$

to ensure that the output of the LSTM unit corresponds to the desired output.

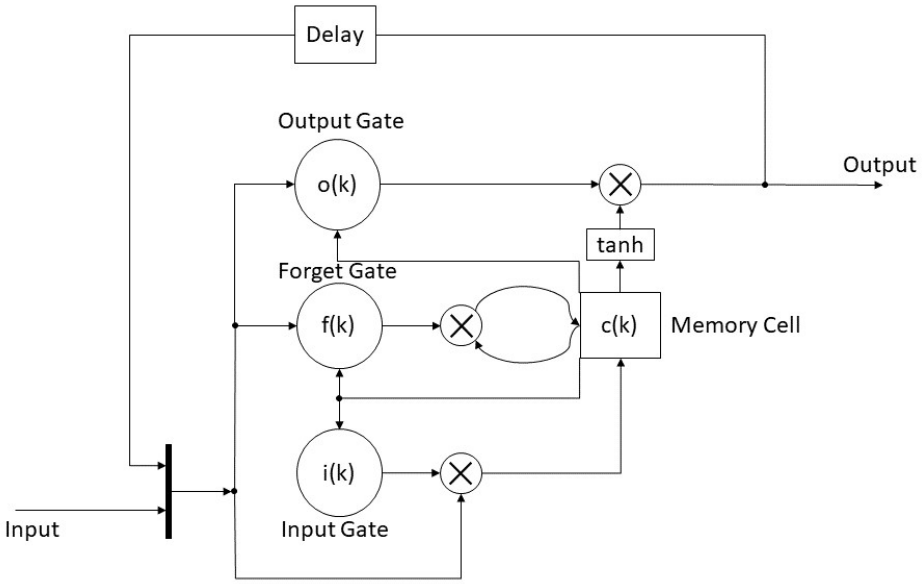

Fig. 1 General form of a LSTM NN unit.

IV. Methodology

A. Quadcopter Data Generation

We use data from the model described in Section II to train a LSTM NN to estimate wind velocities given the trajectory of a quadcopter. In this paper, we focus our efforts on estimating horizontal wind velocities. The inputs to the NN are the roll and pitch angles (ϕ and θ) of the quadcopter and its position errors in the north and east directions (e_{p_n} and e_{p_e}) and the targets for the NN are the wind velocities (V_w) in the north and east directions.

For training, validation, and test data, we generate 5000 seconds of flight data using the Simulink quadcopter simulation with the parameters given in Table 1. The quadcopter is given a waypoint far enough directly north of its location to ensure pitch saturation throughout the simulation. Uniformly distributed random north and east wind disturbances between -7 and 7 m/s are applied for a random amount of time between 0 and 15 seconds each and data are recorded at 10 Hz. North and east position errors and wind velocities are recorded as experienced by the quadcopter. These data are then used to train a LSTM NN using Keras [18] in Python.

After the neural network is trained, the wind velocity outputs for trajectory inputs not in the training or validation datasets are compared to the true values from the Simulink quadcopter model. Wind estimation errors are calculated for 3000 seconds of flight data. The resulting error histograms and a subset of the estimated and actual wind velocities are discussed in Section V.

B. Neural Network Training

NN training results can vary significantly depending on the choice of loss function and optimizer. Common loss function choices include mean square error (MSE), mean absolute error, and mean squared error. Since we focus on

Dynamics parameters							
g	m	J_x	J_y	J_z	L	k_1, k_2	K_f
9.81	1.5	0.0348	0.0459	0.0977	0.235	5×10^{-5}	0.003

Position control gains		
k_p	k_d	k_i
0.3	0.25	0.0002

Attitude control gains					
K_1, K_2	K_3	K_{p1}, K_{p2}	K_{p3}	K_{d1}, K_{d2}	K_{d3}
21.93	48	4.65	3.77	0.1872	0.1496

Table 1 Quadcopter model parameters.

the RMSE of the predicted values, MSE is the intuitive choice. Regarding optimizers, there are a number of common options, such as Stochastic Gradient Descent (SGD) variants, RMSPROP, and Adaptive Moment Estimation (Adam) [19]. We tested RMSPROP and Adam since they have been shown to have excellent convergence properties and are available in Keras. Adam with a learning rate of 0.001 and a batch size of 10 resulted in the smallest losses for our problem and thus is the optimizer that we use to train the NNs for this paper.

We train the NN on 1800 seconds of data with a sequence length of 10 inputs and an overlap of 5 previous inputs with the previous training sequences. Ten percent of the sequences are randomly chosen as the validation sequences for training purposes. For the size of the neural network, we choose to use two hyperbolic tangent hidden layers with 100 units each and 20 percent dropout. This results in low training and validation losses and has a relatively short training time (around 6 minutes) on a NVIDIA GeForce GTX 745.

V. Results

A. Waypoint Navigation Controller

1. Constant Winds

We use the methodology discussed in Section IV-A to train a LSTM NN, resulting in the loss shown in Figure 2. As can be seen from the plot of wind velocity estimates in Figure 3 and the estimation error histogram in Figure 4, this NN provides accurate steady-state estimates of the wind velocities in the test dataset.

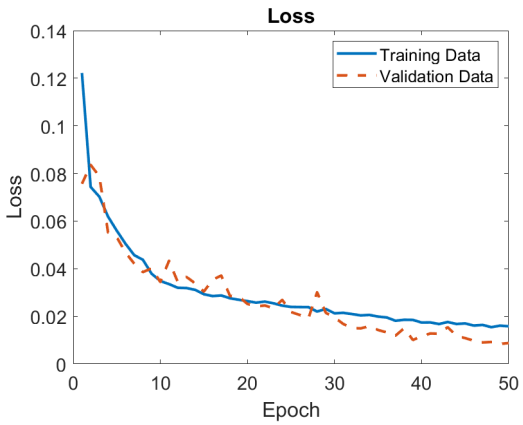

Fig. 2 Training loss for the wind estimation NN for the waypoint navigation controller.

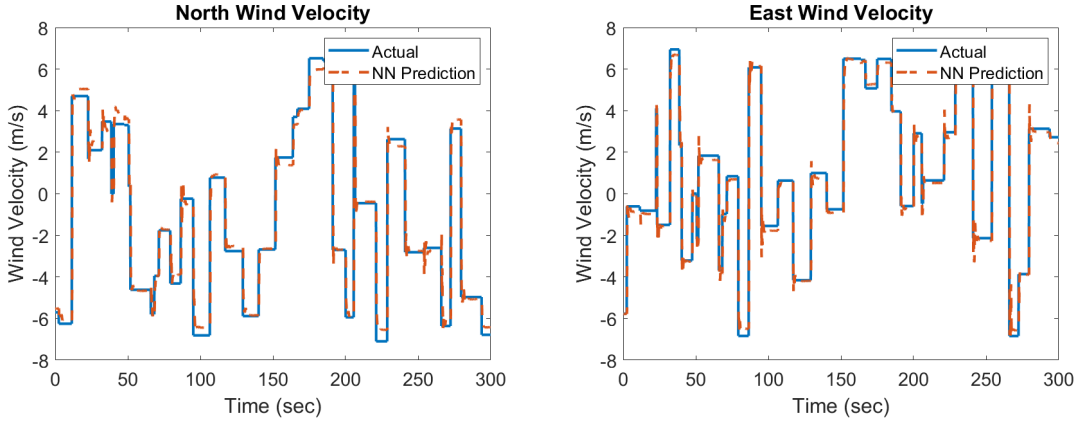

Fig. 3 Comparison of the true wind velocity and the NN predicted wind velocity using the test dataset for the waypoint navigation controller.

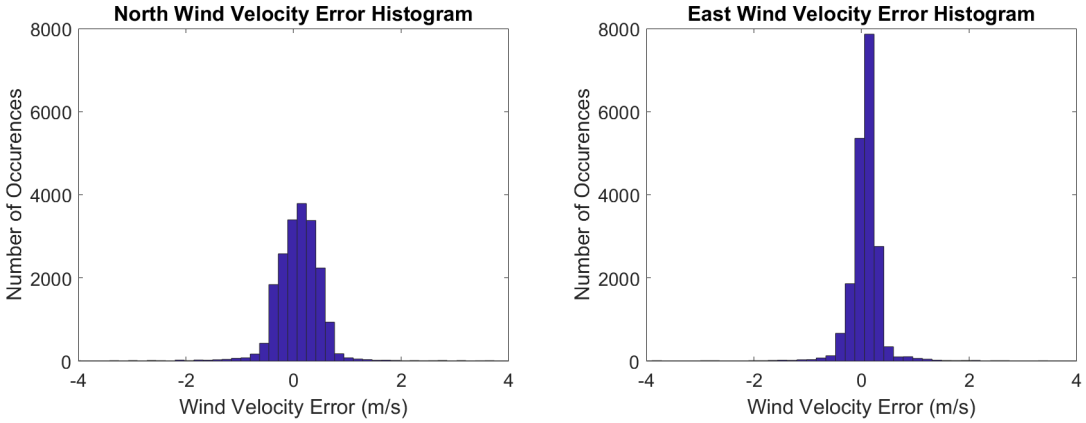

Fig. 4 Wind prediction error histograms for the test dataset with the waypoint navigation controller. Note that wind velocity errors greater than 4 m/s occurred near the large instantaneous jumps in wind velocity but were not shown here.

2. Dryden Winds

After testing the wind estimation of the trained NN using the constant wind, we generate a new dataset using winds from the Dryden model [20]. The Dryden wind model filters a constant wind to result in a simulated wind with realistic frequency components and a mean equal to the constant wind. This simulated wind is generated in the body frame and then rotated to the inertial frame and added to the quadcopter dynamics. In the Dryden model, we set the turbulence intensity to $\sigma = [1.06, 1.06, 0.7]^T$ and the scale length to $L = [200, 200, 50]^T$, respectively. As in the constant wind case, the mean value of the wind has instantaneous jumps between -7 and 7 m/s.

After generating the simulation result of the quadcopter trajectory with the Dryden wind, we use the NN to estimate the Dryden wind velocities using quadcopter position and Euler angle inputs. The wind estimates and the truth are shown in Figure 5. The errors are slightly larger for the Dryden wind estimation than for the constant wind case, as shown in Figure 6, but are still within ten percent of the actual value most of the time.

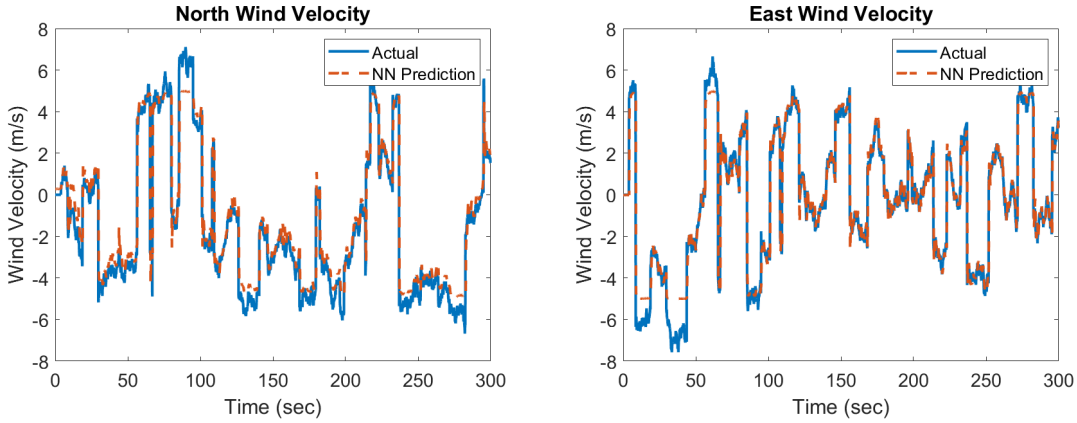

Fig. 5 Comparison of the true wind velocity and the NN predicted wind velocity using the test dataset for the waypoint navigation controller using Dryden wind with a randomly varying mean on $[-7, 7]$ m/s with turbulence intensity $\sigma = [1.06, 1.06, 0.7]^T$ and scale length $L = [200, 200, 50]^T$.

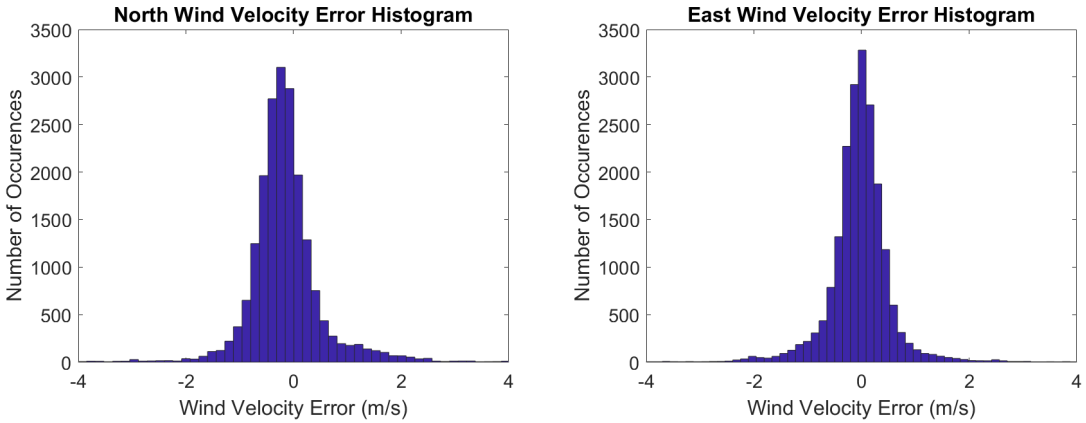

Fig. 6 Wind prediction error histograms for the waypoint navigation controller with Dryden wind. Note that wind velocity errors greater than 4 m/s occurred near the large instantaneous jumps in wind velocity but were not shown here.

B. Geometric Tracking Controller Wind Estimation

1. Constant Winds

To ensure that the modeling approach would work with controllers other than the PD/PID waypoint navigation controller, we generate simulation results using a quadcopter with the geometric tracking controller developed in [21]. We train a NN using the same approach as for the waypoint navigation controller, and use it to predict the wind velocities. As can be seen from Figures 8 and 9, the results are quantitatively similar to those obtained using the waypoint navigation controller.

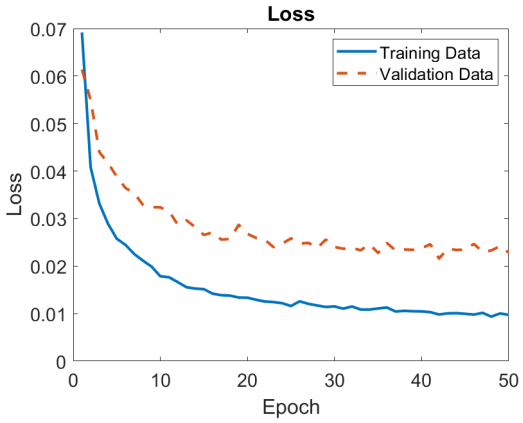

Fig. 7 Training loss for the wind estimation NN for the geometric tracking controller.

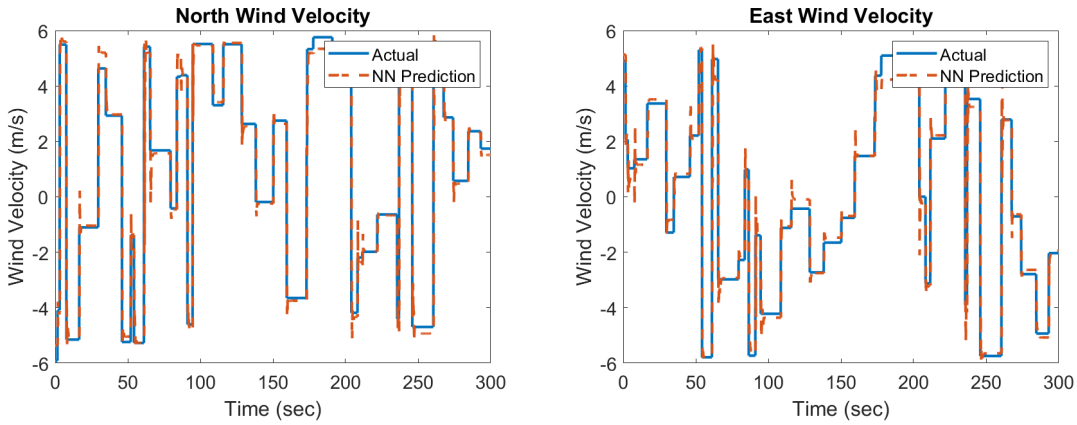

Fig. 8 Comparison of the true wind velocity and the NN predicted wind velocity using the test dataset for the geometric tracking controller.

In Table 2, we present a comparison of the wind estimation errors between the waypoint navigation controller and the geometric tracking controller with constant winds. Errors are calculated with respect to the highest value in each of the considered segments. Although the root mean squared errors (RMSE) and root mean squared percent errors (RMSPE) are similar for both controllers, the tracking controller has slightly smaller errors.

2. Dryden Winds

After ensuring that the NN for the geometric tracking controller accurately estimates the wind velocity for the constant wind case, we generate winds using the same procedure as for the Dryden wind in the waypoint controller case. We then simulate the quadcopter with the tracking controller with the Dryden wind, collect the data, and estimate wind velocities using the NN. As can be seen from Figure 10, the resulting wind estimation follows the general trends of the actual wind. However, this estimation has significantly more error than the constant wind case for the trained NN, as

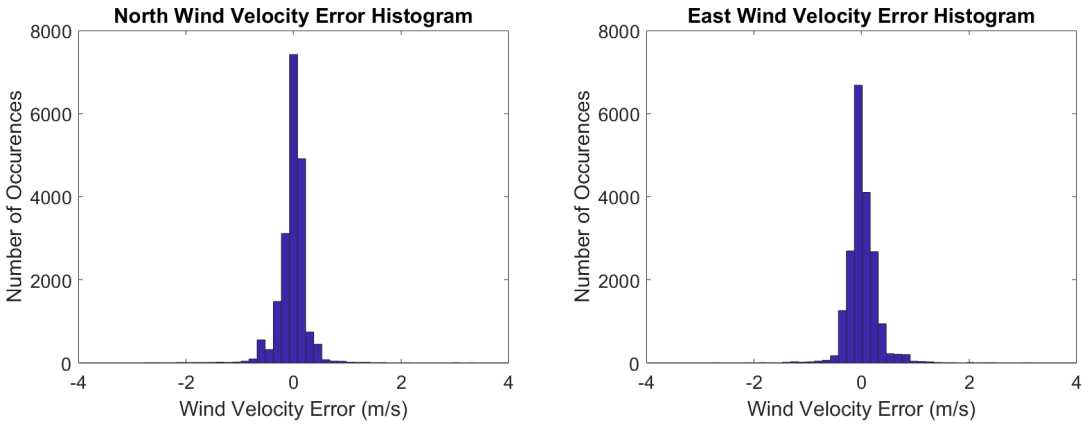

Fig. 9 Wind prediction error histograms for the geometric tracking controller using the test dataset. Note that wind velocity errors greater than 4 m/s occurred near the large instantaneous jumps in wind velocity but were not shown here.

Neural Network Wind Estimation Errors						
	RMSE	RMSPE				
Wind Speed	0-10 m/s	0-10 m/s	0-2.5 m/s	2.5-5 m/s	5-7.5 m/s	7.5-10 m/s
Waypoint Navigation Controller	0.56 m/s	5.65%	27.10%	9.99%	6.59%	7.71%
Geometric Tracking Controller	0.53 m/s	5.34%	26.04%	9.09%	7.58%	6.20%

Table 2 Comparison of wind velocity estimation errors of the NN for constant winds between the waypoint navigation controller and the geometric tracking controller. Percent errors for the NN are calculated with respect to the highest allowed value in each set.

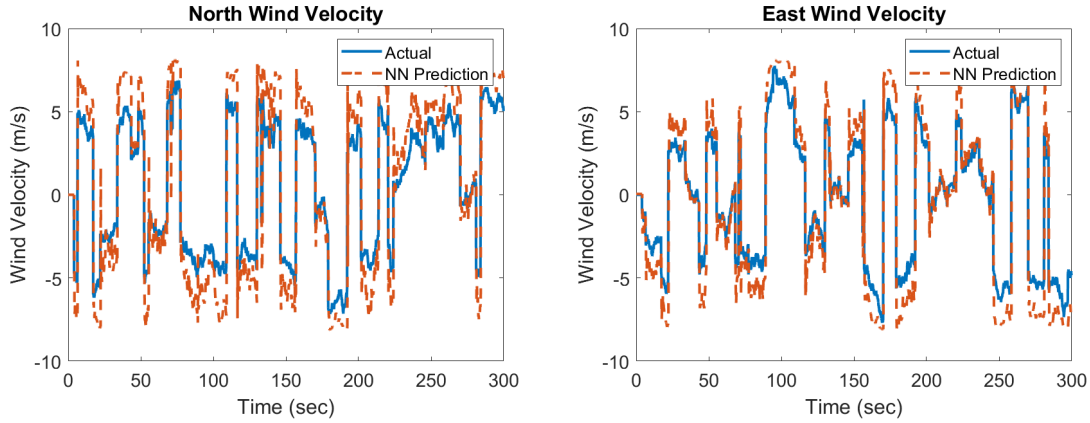

Fig. 10 Comparison of the true wind velocity and the NN predicted wind velocity using the Dryden wind model for the geometric tracking controller.

shown by Figure 11. The majority of this error can be reduced by training the NN on turbulent wind data. This results in error histograms similar to what is seen in the waypoint navigation controller case for the Dryden wind.

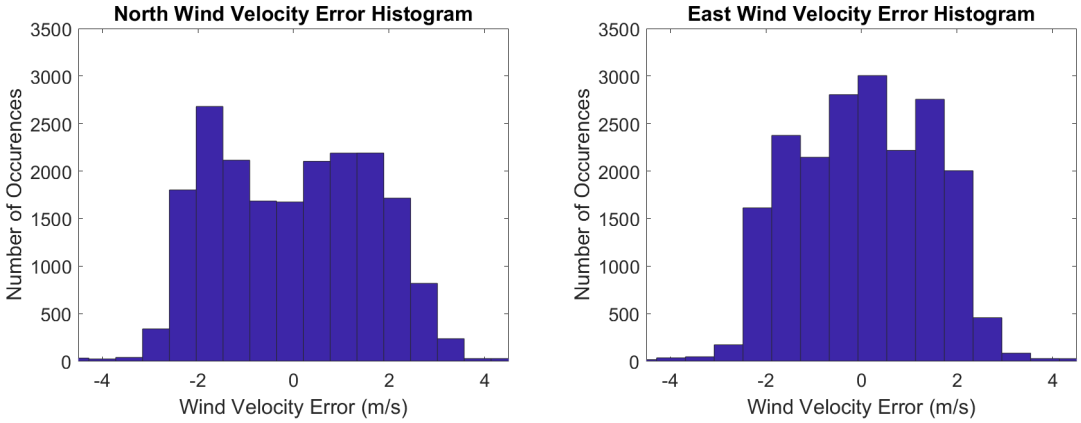

Fig. 11 Wind prediction error histograms for the geometric tracking controller using the Dryden wind model. Note that wind velocity errors greater than 4 m/s occurred near the large instantaneous jumps in wind velocity but were not shown here.

VI. Conclusions and Future Work

We have discussed a neural network approach to estimating wind velocities based on quadcopter trajectories. To validate the approach, we train and test LSTM NNs on simulation data using a waypoint navigation controller and a geometric tracking controller. We manually optimize the hyperparameters of the NN to achieve accurate results. As shown in our simulation studies, the resulting estimations are typically within 5-6% of the maximum measured value for the constant wind case. After verifying that the NNs can estimate constant winds, we test the trained NNs on data generated by flying the quadcopter through simulated turbulent winds. The NNs are still capable of estimating the wind velocities.

Future work includes validating the proposed approach using experimental data, training NNs for quadcopter trajectories other than the straight line case, and comparing the results of the LSTM NN to other types of NNs.

Acknowledgment

We acknowledge financial support from OK NASA EPSCoR Research Initiation Grant and Oklahoma State University (OSU) research start-up. The authors also thank Dr. Min Xue from NASA Ames and Dr. Marty Hagan from OSU for many helpful discussions on trajectory modeling and neural networks.

References

[1] Allison, S., Bai, H., and Jayaraman, B., "Modeling Trajectory Performance of Quadrotors Under Wind Disturbances," *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 1237.

[2] Narendra, K. S., and Parthasarathy, K., "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, 1990, pp. 4–27. doi:10.1109/72.80202.

[3] CHEN, S., and BILLINGS, S. A., "Neural networks for nonlinear dynamic system modelling and identification," *International Journal of Control*, Vol. 56, No. 2, 1992, pp. 319–346. doi:10.1080/00207179208934317, URL `https://doi.org/10.1080/00207179208934317`.

[4] Phan, M. C., Beale, M. H., and Hagan, M. T., "A procedure for training recurrent networks," *Neural Networks (IJCNN), The 2013 International Joint Conference on*, IEEE, 2013, pp. 1–8.

[5] Zaremba, W., Sutskever, I., and Vinyals, O., "Recurrent Neural Network Regularization," *CoRR*, Vol. abs/1409.2329, 2014. URL `http://arxiv.org/abs/1409.2329`.

[6] Langelaan, J. W., Alley, N., and Neidhoefer, J., "Wind field estimation for small unmanned aerial vehicles," *Journal of Guidance Control and Dynamics*, Vol. 34, No. 4, 2011, p. 1016.

[7] Waslander, S. L., and Wang, C., "Wind Disturbance Estimation and Rejection for Quadrotor Position Control," *AIAA Infotech@Aerospace Conference*, AIAA, 2009.

[8] Bruschi, P., Piotto, M., Dell'Agnello, F., Ware, J., and Roy, N., "Wind speed and direction detection by means of solid-state anemometers embedded on small quadcopters," *Procedia Engineering*, Vol. 168, 2016, pp. 802–805.

[9] Laurence III, R. J., "sUAS Wind Sensing with Computational Fluid Dynamics and a Distributed Flush Airdata System," Ph.D. thesis, University of Colorado at Boulder, 2017.

[10] Neumann, P. P., and Bartholmai, M., "Real-time wind estimation on a micro unmanned aerial vehicle using its inertial measurement unit," *Sensors and Actuators A: Physical*, Vol. 235, 2015, pp. 300–310.

[11] Rodriguez Salazar, L., Cobano, J. A., and Ollero, A., "Small uas-based wind feature identification system part 1: Integration and validation," *Sensors*, Vol. 17, No. 1, 2016, p. 8.

[12] Xue, M., "UAV Trajectory Modeling Using Neural Networks," *17th AIAA Aviation Technology, Integration, and Operations Conference*, AIAA, 2017.

[13] Furrer, F., Burri, M., Achtelik, M., and Siegwart, R., *Robot Operating System (ROS): The Complete Reference (Volume 1)*, Springer International Publishing, Cham, 2016, Chaps. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. doi:10.1007/978-3-319-26054-9_23, URL http://dx.doi.org/10.1007/978-3-319-26054-9_23.

[14] Xiang, C., Wang, X., Ma, Y., and Xu, B., "Practical modeling and comprehensive system identification of a BLDC motor," *Mathematical Problems in Engineering*, Vol. 2015, 2015.

[15] Beard, R., "Quadrotor dynamics and control rev 0.1," 2008.

[16] Sydney, N., Smyth, B., and Paley, D. A., "Dynamic control of autonomous quadrotor flight in an estimated wind field," *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, IEEE, 2013, pp. 3609–3616.

[17] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, 2016. http://www.deeplearningbook.org.

[18] Chollet, F., et al., "Keras," https://github.com/fchollet/keras, 2015.

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

[20] Beard, R. W., and McLain, T. W., *Small Unmanned Aircraft - Theory and Practice*, Princeton University Press, 2012.

[21] Lee, T., Leoky, M., and McClamroch, N. H., "Geometric tracking control of a quadrotor UAV on SE (3)," *Decision and Control (CDC), 2010 49th IEEE Conference on*, IEEE, 2010, pp. 5420–5425.