Pretraining with Neural-Fly for Rapid Online Learning

Michael O'Connell[*,1], Guanya Shi[*,1], Xichen Shi[1], Kamyar Azizzadenesheli[2], Animashree Anandkumar[1], Yisong Yue[1], and Soon-Jo Chung[1]

***Abstract*—Executing safe and precise flight maneuvers in dynamic high-speed winds is important for the ongoing commoditization of uninhabited aerial vehicles (UAVs). However, since the relationship between various wind conditions and its effect on aircraft maneuverability is not well understood, it is challenging to design effective robot controllers using traditional control design methods. We present Neural-Fly, a learning-based approach that allows rapid online adaptation by incorporating pre-trained representations through deep learning. Neural-Fly builds on two key observations that aerodynamics in different wind conditions share a common representation and that the wind-specific part lies in a low-dimensional space. To that end, Neural-Fly uses a proposed learning algorithm, Domain Adversarially Invariant Meta-Learning (DAIML), to learn the shared representation, only using 12 minutes of flight data. This pretraining phase enables rapid online learning through a composite adaptation law, which only needs to update a set of linear coefficients for mixing the basis elements to effectively correct for the wind effects. When evaluated under challenging wind conditions generated with the Caltech Real Weather Wind Tunnel with wind speeds up to 43.6 km/h (12.1 m/s), Neural-Fly achieves precise flight control with substantially smaller tracking error than state-of-the-art nonlinear and adaptive controllers. In addition to strong empirical performance, the exponential stability of Neural-Fly results in robustness guarantees. Finally, our control design extrapolates to unseen wind conditions, is shown to be effective for outdoor flights with only on-board sensors, and can transfer across drones with minimal performance degradation.**

I. Introduction

The proliferation of uninhabited aerial vehicles (UAVs) offers the prospect to revolutionize many aspects of our daily lives but requires increased precision and robustness. Applications range from drone delivery to drone rescue and search, and from urban air mobility to autonomous farming tools. Howevever, these applications demand precise and agile control methods that can handle the complex aerodynamics while adapting to changing environmental and operating conditions. Flying in windy environments introduces even more complexity because of the unsteady aerodynamic interactions between the drone, the induced airflow, and the wind. These unsteady and nonlinear aerodynamic effects substantially degrade the performance of conventional UAV control methods that neglect to account for them in the control design. Our recent work, Neural-Fly [1], offers a solution, by pretraining a neural network to enable rapid and robust online learning of wind effects.

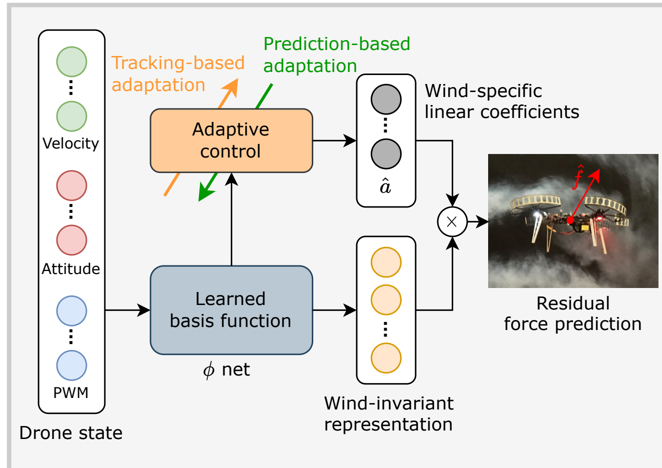

Fig. 1: **Neural-Fly design.** Neural-Fly learns a model of aerodynamics with linearly separated wind-variant and wind-invariant components. A set of meta-trained basis functions, ϕ, is the wind-invariant representation of the aerodynamic effects. A composite adaptation algorithm (that is, including tracking-error-based and prediction-error-based terms) to update wind-specific linear weights $\hat{a}$. The result is the wind-effect force estimate, $\hat{f} = \phi\hat{a}$, which can be quickly adapted to new wind conditions and used for precise control of the UAV in challenging scenarios.

Prior approaches partially capture these effects with simple linear or quadratic air drag models, which limit the tracking performance in agile flight and cannot be extended to external wind conditions [2], [3]. Although more complex aerodynamic models can be derived from computational fluid dynamics [4], such modelling is often computationally expensive, and is limited to steady non-dynamic wind conditions. Adaptive control addresses this problem by estimating linear parametric uncertainty in the dynamical model in real time to improve tracking performance. Recent state-of-the-art in quadrotor flight control has used adaptive control methods that directly estimate the unknown aerodynamic force without assuming the structure of the underlying physics, but relying on high-frequency and low-latency control [5]–[8]. In parallel, there has been increased interest in data-driven modeling of aerodynamics (e.g., [9]–[12]), however existing approaches cannot effectively adapt in changing or unknown environments such as time-varying wind conditions.

[*] Equal contribution to this work. [1]California Institute of Technology, [2]Purdue University.

In this short paper, we discuss the general method and applications of our recently-developed data-driven approach, called Neural-Fly[1], and how it can be used to pretrain learning-based control algorithms. Our method, depicted in Fig. 1, demonstrates an efficient algorithm to pretrain a neural network so that it can be adapted to different environments. Neural-Fly also demonstrates a robust method to update such a neural network in real-time, using our robust adaptation algorithm. Neural-Fly has been applied to deep-learning-based trajectory tracking control, and it has allowed quick adaptation to rapidly-changing wind conditions with centimeter-level position-error tracking of agile manuevers. Furthermore, Neural-Fly has demonstrated the ability to transfer control policies from one robot to another, and from limited range of constant wind speeds to a wide range of time-varying wind speeds.

II. The Neural-Fly Method

Consider the general robot dynamics model

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u + f(q,\dot{q},w) \tag{1}$$

where $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$ are the n dimensional position, velocity, and acceleration vectors, $M(q)$ is the symmetric, positive definite inertia matrix, $C(q,\dot{q})$ is the Coriolis matrix, $g(q)$ is the gravitational force vector and $u \in \mathbb{R}^n$ is the control force. Most importantly, $f(q,\dot{q},w)$ incorporates unmodeled dynamics, and $w \in \mathbb{R}^m$ is an unknown hidden state used to represent the underlying environmental conditions, which is potentially time-variant. Specifically, w represents the wind profile, and each different wind profile yields different unmodeled aerodynamic disturbances for the UAV.

The Neural-Fly algorithm decomposes the unmodeled dynamics into a wind-condition-independent basis function $\phi(q,\dot{q})$ and a wind-condition-dependent linear coefficient $a(w)$, that is,

$$f(q,\dot{q},w) \approx \phi(q,\dot{q})a(w). \tag{2}$$

In the supplementary material for [1], we provided that the decomposition $\phi(q,\dot{q})a(w)$ exists for any analytic function $f(q,\dot{q},w)$, analyze ability of our method to untangle the dependance of ϕ on w, and demonstrate the stability and robustness of the Neural-Fly adaptation algorithm and overall method through stability analysis and experimental demonstrations. Here, we will provide an overview of the algorithms and provide some intuition for the key aspects that allow Neural-Fly to transfer to new wind conditions and vehicles.

Our method has two main stages: an offline learning phase and an online adaptive control phase used as real-time online learning. For the offline learning phase, we have developed Domain Adversarially Invariant Meta-Learning (DAIML) that learns a wind-condition-independent deep neural network (DNN) representation of the aerodynamics in a data-efficient manner. The full algorithm is given in **??** 1. The output of the DNN is treated as a set of basis functions that represent the aerodynamic effects. This representation is adapted to different wind conditions by updating a set of linear coefficients that mix the output of the DNN. DAIML is data efficient and uses only 12 total minutes of flight data in just 6 different wind conditions to train the DNN. DAIML incorporates several key features which not only improve the data efficiency but also are informed by the downstream online adaptive control phase. In particular, DAIML uses spectral normalization [9], [13] to control the Lipschitz property of the DNN to improve generalization to unseen data and provide closed-loop stability and robustness guarantees. As seen in Fig. 2, training data generated in different wind conditions can have high correlation between the actual trajectory of the vehicle and the wind condition present. To counter this correlation and prevent overfitting, DAIML uses a discriminative network, which ensures that the learned representation is wind-invariant and that the wind-dependent information is only contained in the linear coefficients that are adapted in the online control phase. The result is that DAIML trains a concise representation of the aerodynamics that is both data efficient and generalizes well to new wind conditions and even new vehicles.

Algorithm 1: Domain Adversarially Invariant Meta-Learning (DAIML)

Hyperparameter: $\alpha \geq 0$, $0 < \eta \leq 1$, $\gamma > 0$
Input: $\mathcal{D} = \{D_{w_1}, \cdots, D_{w_K}\}$
Initialize: Neural networks ϕ and h
Result: Trained neural networks ϕ and h

repeat
  Randomly sample D_{w_k} from $\mathcal{D}$
  Randomly sample two disjoint batches B^a (adaptation set) and B (training set) from D_{w_k}
  Solve the least squares problem
  $$a^*(\phi) = \arg\min_a \sum_{i \in B^a} \left\| y_k^{(i)} - \phi(x_k^{(i)})a \right\|^2$$
  if $\|a^*\| > \gamma$ **then**
    $a^* \leftarrow \gamma \cdot \frac{a^*}{\|a^*\|}$ ▷ `normalization`
  Train DNN ϕ using stochastic gradient descent (SGD) and spectral normalization with loss
  $$\sum_{i \in B} \left(\left\| y_k^{(i)} - \phi(x_k^{(i)})a^* \right\|^2 - \alpha \cdot \text{loss}\left(h(\phi(x_k^{(i)})), k \right) \right)$$
  if `rand()` $\leq \eta$ **then**
    Train DNN h using SGD with loss
    $\sum_{i \in B} \text{loss}\left(h(\phi(x_k^{(i)})), k \right)$
until convergence

For the online adaptive control phase, we have developed a regularized composite adaptive control law to enable fast and robust adaptation to new wind conditions. The adaptation algorithm is built from a Kalman Filter [14], [15] estimator of the linear coefficients, $a(w)$. The underlying model used in the Kalman Filter design naturally provides robustness and regularization properties. The adaptation law derived from the Kalman Filter is augmented with a tracking error term to make the closed loop dynamics more stable during rapid adaptation.

The adaptation law updates the wind-dependent linear coefficients using a composite of the position tracking error term and the aerodynamic force prediction error term. This approach effectively guarantees stable and fast adaptation to any wind condition and robustness against imperfect learning. Although this adaptive control law could be used with a number of learned models, the speed of adaptation is further aided by the concise representation learned from DAIML.

The Neural-Fly adaptive control algorithm can be summarized by the following control law, adaptation law, and covariance update equations, respectively.

$$u_{\mathrm{NF}} = \underbrace{M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + g(q)}_{\text{nominal model feedforward terms}} \underbrace{-Ks}_{\text{PD feedback}} \underbrace{-\phi(q,\dot{q})\hat{a}}_{\text{learning-based feedforward}} \tag{3}$$

$$\dot{\hat{a}} = \underbrace{-\lambda\hat{a}}_{\text{regularization term}} \underbrace{-P\phi^\top R^{-1}(\phi\hat{a}-y)}_{\text{prediction error term}} \underbrace{+P\phi^\top s}_{\text{tracking error term}} \tag{4}$$

$$\dot{P} = -2\lambda P + Q - P\phi^\top R^{-1}\phi P \tag{5}$$

where u_{NF} is the control law, $\dot{\hat{a}}$ is the online linear-parameter update, P is a covariance-like matrix used for automatic gain tuning, $s = \dot{\tilde{q}} + \Lambda\tilde{q}$ is the composite tracking error, y is the measured aerodynamic residual force with measurement noise ϵ, and K, Λ, R, Q, and λ are gains.

A key result of the Neural-Fly method is robustness to error in the learned representation of the unmodeled dynamics. Here, we provide a brief overview of the stability and robustness guarantees for the Neural-Fly method. First, we formally define the representation error $d(t)$, as the difference between the unknown dynamics $f(q,\dot{q},w)$ and the best linear weight vector a given the learned representation $\phi(q,\dot{q})$, namely, $d(t) = f(q,\dot{q},w) - \phi(q,\dot{q})a(w)$. The measurement noise for the measured residual force is a bounded function $\epsilon(t)$ such that $y(t) = f(t) + \epsilon(t)$. If the environment conditions are changing, we consider the case that $\dot{a} \neq 0$. This leads to the following stability theorem.

Theorem 1. *If we assume that the desired trajectory has bounded derivatives and the system evolves according to the dynamics in Eq.* (1)*, the control law Eq.* (3)*, and the adaptation law Eq.* (4) *and* (5)*, then the position tracking error exponentially converges to the ball*

$$\lim_{t\to\infty} \|\tilde{q}\| \leq \sup_t \left[C_1\|d(t)\| + C_2\|\epsilon(t)\| + C_3\left(\lambda\|a(t)\| + \|\dot{a}(t)\|\right)\right], \tag{6}$$

where C_1, C_2, and C_3 are three bounded constants depending on $\phi, R, Q, K, \Lambda, M$ and λ.

The proof of this theorem is provided in the supplementary material for [1].

III. Results

We built a quadrotor UAV for our primary data collection and all experiments, shown flying through narrow gates with wind and smoke in Fig. 3. This vehicle features a wide-X configuration, weighs 2.6 kg, tilted motors, and is built off standard flight control software, PX4, and standard robotic middleware, Robotic Operating System.

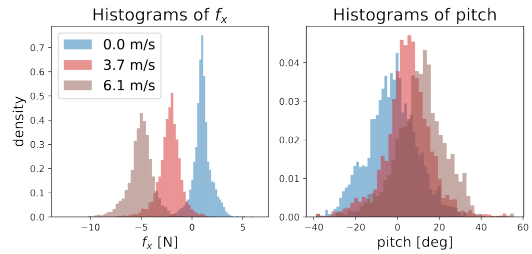

Fig. 2: **Input-output correlation in the training data.** Histograms showing data distributions in different wind conditions. **Left:** distributions of the x-component of the wind-effect force, f_x. This shows that the aerodynamic effect changes as the wind varies. **Right:** distributions of the pitch, a component of the state used as an input to the learning model. This shows that the shift in wind conditions causes a distribution shift in the input.

To study the generalizability and robustness of our approach, we also use an Intel Aero Ready to Fly drone to collect an alternate dataset. This dataset is used to train a representation of the wind effects on the Intel Aero drone, which we test on our custom UAV. The Intel Aero drone has a symmetric X configuration, weighs 1.4 kg, and does not have tilted motors.

Neural-Fly was tested on an agile figure-8 trajectory and compared with several methods that represent the state of art in quadrotor control. Each method was tested in a variety of wind conditions, including wind speeds inside the range of wind speeds seen in training (0 m/s to 4.2 m/s), and wind speeds outside the range of wind speeds seen in training (8.5 m/s to 12.1 m/s), and time varying wind speeds ($8.5 + 2.4\sin(t)$m/s) that break the constant wind assumption made during training. Using Neural-Fly, we report an average improvement of 66 % over a nonlinear tracking controller [16], 42 % over an $\mathcal{L}_1$ adaptive controller [8], and 35 % over an Incremental Nonlinear Dynamics Inversion (INDI) controller [5]. These results are all accomplished using standard quadrotor UAV hardware, while running the PX4's default regulation attitude control. Our tracking performance is competitive even compared to related work without external wind disturbances and with more complex hardware (for example, [5] requires a 10-time higher control frequency and onboard optical sensors for direct motor speed feedback).

We also compare Neural-Fly with two variants of our method: Neural-Fly-Transfer, which uses a learned representation trained on data from a the Intel-Aero drone, and Neural-Fly-Constant, which only uses our adaptive control law with a trivial non-learning basis. Neural-Fly-Constant, $\mathcal{L}_1$, and INDI all directly adapt to the unknown dynamics without assuming the structure of the underlying physics, and they have similar performance. Neural-Fly-Transfer demonstrates that our method is robust to changes in vehicle configuration and model mismatch. This robustness is a key advantage of

TABLE I: Tracking error statistics in cm for different wind conditions. Two metrics are considered: root-mean-square (RMS) and mean.

Method \ Wind speed [m/s]	0		4.2		8.5		12.1		$8.5 + 2.4\sin(t)$	
	RMS	Mean	RMS	Mean	RMS	Mean	RMS	Mean	RMS	Mean
Nonlinear	11.9	10.8	10.7	9.9	16.3	14.7	23.9	21.6	31.2	28.2
INDI	7.3	6.3	6.4	5.9	8.5	8.2	10.7	10.1	11.1	10.3
L1	4.6	4.2	5.8	5.2	12.1	11.1	22.7	21.3	13.0	11.6
NF-Constant	5.4	5.0	6.1	5.7	7.5	6.9	12.7	11.2	12.7	12.1
NF-Transfer	3.7	3.4	4.8	4.4	6.2	5.9	10.2	9.4	8.8	8.0
NF	**3.2**	**2.9**	**4.0**	**3.7**	**5.8**	**5.3**	**9.4**	**8.7**	**7.6**	**6.9**

Fig. 3: **Agile flight through narrow gates.** Neural-Fly was tested in the Caltech Real Weather Wind Tunnel where wind effects can be visualized using smoke machines. The UAV follows an agile trajectory through narrow gates, which are slightly wider than the UAV itself, under challenging wind conditions. These panels show the moment the UAV passed through the gate and the complex interaction between the UAV and the wind.

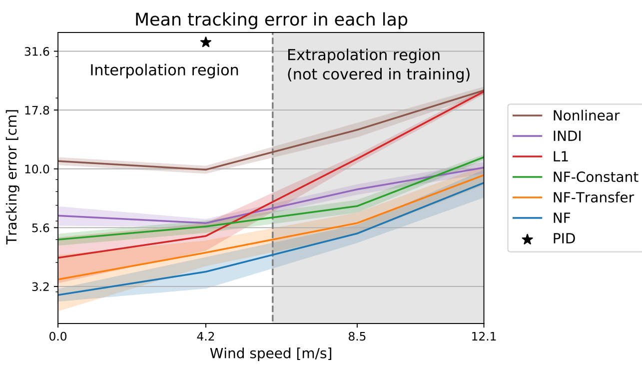

Fig. 4: **Mean tracking errors in different wind conditions.** Solid lines show the mean error over 6 laps and the shade areas show standard deviation of the mean error on each lap.

our method, which can be used to control a wide range of quadrotor UAVs without requiring a new model to be trained for each vehicle.

Finally, we demonstrate that our method enables a new set of capabilities that allow the UAV to fly through low-clearance gates following agile trajectories in gusty wind conditions (Fig. 3).

Together, these tests demonstrate not only the effectiveness of our method, but also its robustness to modeling error and generalization to new conditions, key considerations for pretraining adaptable controllers.

IV. Conclusion

When measuring position tracking errors, we observe that our Neural-Fly method outperforms state-of-the-art flight controllers in all wind conditions. Neural-Fly can generalize to new conditions, as demonstrated by its performance in wind speeds outside the training range and in time varying wind speeds. Furthermore, Neural-Fly is robust to changes in vehicle configuration and modeling errors, as demonstrated by the similar performance of Neural-Fly-Transfer. Our control algorithm is formulated generally for all robotic systems described by the Euler-Langrange equation, and should be applicable to a wide range of robotic systems. Neural-Fly demonstrates a new paradigm for designing adaptable controllers that can be trained once and then used to control a wide range of vehicles.

Acknowledgements

A.A. is also affiliated with NVIDIA Corporation, and Y.Y. is also with associated Argo AI. We thank J. Burdick and J.-J. E. Slotine for their helpful discussions. We thank M. Anderson for his help configuring the quadrotor platform, and M. Anderson and P. Spieler for their help troubleshooting hardware. We also thank N. Badillo and L. Pabon Madrid for help in experiments.

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). This research was also conducted in part with funding from Raytheon Technologies. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. The experiments reported in this article were conducted at Caltech's Center for Autonomous Systems and Technologies (CAST).

References

[1] M. O'Connell, G. Shi, X. Shi, *et al.*, "Neural-Fly enables rapid learning for agile flight in strong winds," *Science Robotics*, May 4, 2022. DOI: 10.1126/scirobotics.abm6597. [Online]. Available: https://www.science.org/doi/full/10.1126/scirobotics.abm6597 (visited on 05/13/2022).

[2] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, Jul. 21, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abh1221 (visited on 09/08/2021).

[3] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, Apr. 2018, ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2776353.

[4] P. Ventura Diaz and S. Yoon, "High-Fidelity Computational Aerodynamics of Multi-Rotor Unmanned Aerial Vehicles," in *2018 AIAA Aerospace Sciences Meeting*, ser. AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, Jan. 7, 2018. DOI: 10.2514/6.2018-1266. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2018-1266 (visited on 03/27/2023).

[5] E. Tal and S. Karaman, "Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1203–1218, May 2021, ISSN: 1558-0865. DOI: 10.1109/TCST.2020.3001117.

[6] S. Mallikarjunan, B. Nesbitt, E. Kharisov, E. Xargay, N. Hovakimyan, and C. Cao, "L1 Adaptive Controller for Attitude Control of Multirotors," in *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, Minnesota: American Institute of Aeronautics and Astronautics, Aug. 13, 2012, ISBN: 978-1-60086-938-9. DOI: 10.2514/6.2012-4831. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2012-4831 (visited on 03/04/2022).

[7] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou, "$\Mathcal{L}$1-Adaptive MPPI Architecture for Robust and Agile Control of Multirotors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 7661–7666. DOI: 10.1109/IROS45743.2020.9341154.

[8] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, Precision, and Payloads: Adaptive Nonlinear MPC for Quadrotors," Sep. 9, 2021. arXiv: 2109.04210 [cs]. [Online]. Available: http://arxiv.org/abs/2109.04210 (visited on 09/16/2021).

[9] G. Shi, X. Shi, M. O'Connell, *et al.*, "Neural Lander: Stable Drone Landing Control using Learned Dynamics," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9784–9790, May 2019. DOI: 10.1109/ICRA.2019.8794351. arXiv: 1811.08027. [Online]. Available: http://arxiv.org/abs/1811.08027 (visited on 09/02/2021).

[10] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung, "Neural-Swarm: Decentralized Close-Proximity Multirotor Control Using Learned Interactions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 3241–3247. DOI: 10.1109/ICRA40945.2020.9196800.

[11] G. Shi, W. Hönig, X. Shi, Y. Yue, and S.-J. Chung, "Neural-Swarm2: Planning and Control of Heterogeneous Multirotor Swarms Using Learned Interactions," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1063–1079, Apr. 2022, ISSN: 1941-0468. DOI: 10.1109/TRO.2021.3098436.

[12] G. Torrente, E. Kaufmann, P. Fohn, and D. Scaramuzza, "Data-Driven MPC for Quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, Apr. 2021, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3061307. [Online]. Available: https://ieeexplore.ieee.org/document/9361343/ (visited on 10/08/2021).

[13] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/b22b257ad0519d4500539da3c8bcf4dd-Abstract.html (visited on 11/18/2022).

[14] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1, 1960, ISSN: 0021-9223. DOI: 10.1115/1.3662552. [Online]. Available: https://doi.org/10.1115/1.3662552 (visited on 09/21/2021).

[15] R. E. Kalman and R. S. Bucy, "New Results in Linear Filtering and Prediction Theory," *Journal of Basic Engineering*, vol. 83, no. 1, pp. 95–108, Mar. 1, 1961, ISSN: 0021-9223. DOI: 10.1115/1.3658902. [Online]. Available: https://asmedigitalcollection.asme.org/fluidsengineering/article/83/1/95/426820/New-Results-in-Linear-Filtering-and-Prediction (visited on 09/21/2021).

[16] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.