Transferable Learning-Based Control using Neural-Fly

Michael O'Connell[*,1], Guanya Shi[*,2], Xichen Shi[1], Kamyar Azizzadenesheli[3], Animashree Anandkumar[1,3], Yisong Yue[1,4], 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 that transfer well to different environments and conditions. Neural-Fly is a learning-based approach that allows rapid online learning by incorporating pre-trained representations through deep learning. Neural-Fly is robust and demonstrates transferability to different tasks, environments, and drones. In this workshop, we will present the latest results for Neural-Fly, discuss implementation considerations, and show case other applications of Neural-Fly.**

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. However, 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. This enables Neural-Fly to transfer between different tasks, environments, and drones while maintaining high performance.

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.

In this extended abstract, 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 train transferable learning-based control algorithms. We will present implementation considerations and show case other applications of Neural-Fly. Our method, depicted in Fig. 1, demonstrates an efficient algorithm to pretrain a neural net-

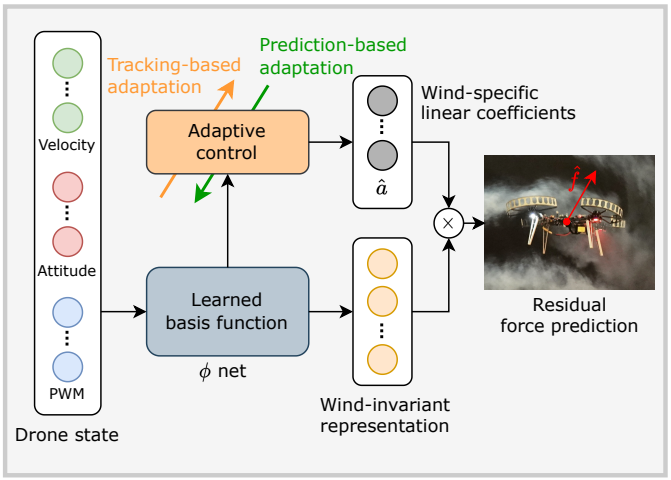

Fig. 1: **Neural-Fly design.** Neural-Fly learns a model of aerodynamics with linearly separated wind-variant and wind-invariant components. Since only part of the model must be updated in real time, Neural-Fly can quickly learn and adapt to new wind conditions.

[*] Equal contribution to this work. [1] California Institute of Technology, [2] Robotics Institute, Carnegie Mellon University, [3] NVIDIA Corporation, [4] Latitude 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).

Fig. 2: **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.

work 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. In this work, w represents the wind profile and vehicle used for training, 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 dependence 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 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 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. 3, 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

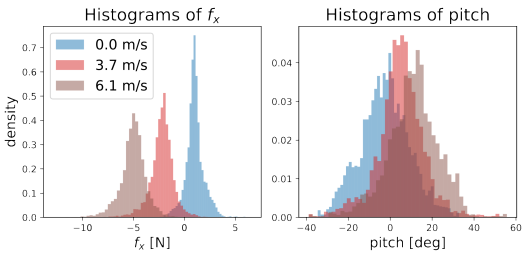

Fig. 3: **Input-output correlation in the training data.** Histograms showing data distributions in different wind conditions, showing that the shift in wind conditions causes a distribution shift in the input **Left:** distributions of the x-component of the wind-effect force, f_x. **Right:** distributions of the pitch, a component of the state used as an input to the learning model.

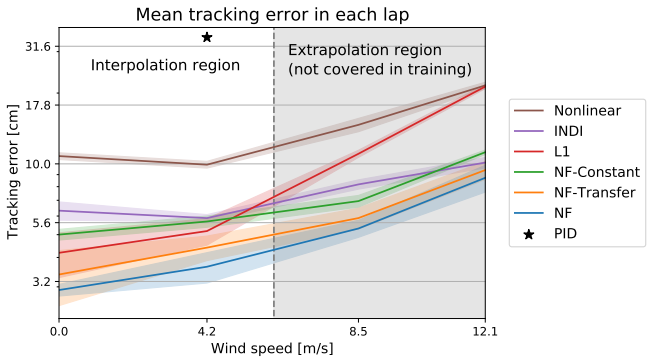

Fig. 4: **Mean tracking errors in different wind conditions.** Solid lines show the mean error over 6 laps and the shade areas show standard deviation of the mean error on each lap.

aerodynamics that is both data efficient and generalizes well to new wind conditions and even new vehicles.

For the online adaptive control phase, Neural-Flyuses a robust and fast adaptive control law to update the model for 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 Kalman Filter is augmented with a tracking error term to make the closed loop dynamics stable during rapid adaptation. The combination of the prediction error based Kalman filter and tracking error based adaptation term makes this approach a composite adaptive control law, and effectively guarantees fast and stable adaptation to any wind condition and robustness against imperfect learning. The speed of adaptation is further aided by the concise representation learned from DAIML.

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, 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, $d(t) = f(q,\dot{q},w) - \phi(q,\dot{q})a(w)$. In [1] Theorem 1, we showed that the upper bound for the vehicle tracking error, $\tilde{q}$, scales linearly with the representation error, namely, $\lim_{t\to\infty} \|\tilde{q}(t)\| \leq C\|d(t)\|$, where C is a constant that depends on the control and adaptation gains. Thus, when transferring to a new task, environment, or vehicle, the Neural-Fly method's performance is bounded by how well the learned representation generalizes to the new task, environment, or vehicle. DAIML improves the generalization of the learned representation, and thus directly improves the transferability of the Neural-Fly method.

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. 2. 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.

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-speed 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 [6], [8], and 35 % over an Incremental Nonlinear Dynamics Inversion (INDI) controller [5].

We also compare Neural-Fly with two variants of our method. The first variant of our method, Neural-Fly-Constant, is similar in structure and performance to $\mathcal{L}_1$ and INDI, all of which directly adapt to the unknown dynamics without assuming the structure of the underlying physics. The second variant, Neural-Fly-Transfer, demonstrates that our method is robust to changes in vehicle configuration and model mismatch. This generalizability and robustness is a key advantage of our method, and suggests that Neural-Fly can be trained once for a class of vehicles and safely transfer to any vehicle in that class.

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. 2).

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 transfering learning-based control algorithms between tasks, vehicles, and environments.

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-Lagrange 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.

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. (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. (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. (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. (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. (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. (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. (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.