# Sim-to-(Multi)-Real: Transfer of Low-Level Robust Control Policies to Multiple Quadrotors

Artem Molchanov*, Tao Chen*, Wolfgang Hönig, James A. Preiss, Nora Ayanian and Gaurav S. Sukhatme

***Abstract*— We use reinforcement learning to train policies in simulation that transfer remarkably well to multiple different physical quadrotors. Our policies are low-level, i.e., we map the rotorcrafts' state directly to the motor outputs. We show how different training methodologies (change of the cost function, modeling of noise, use of domain randomization) might affect flight performance. To the best of our knowledge, this is the first work that demonstrates that a simple neural network can learn a low-level quadrotor controller without the use of a stabilizing PD controller; as well as the first work that analyses the transfer capability of a single policy to multiple quadrotors. The video of our experiments can be found at `https://sites.google.com/view/sim-to-multi-quad`.**

## I. Introduction and Related Work

Traditional methods to quadrotor stabilizing control often require careful, model-specific system identification and parameter tuning to succeed. We are interested in finding a single control policy without manual parameter tuning. Such a control policy is useful for testing of new custom-built quadrotors, and as a backup safety controller. In our work, we use reinforcement learning (RL) with simulated quadrotor models to learn a transferable control policy.

Transferring from simulation to reality (S2R) is often used to overcome the issues of safety, and complexity of data collection on robotic systems. Carefully estimating parameters of the real system to achieve a more realistic simulation can improve the transfer quality [1]–[4] but often requires complex setups [4]. An alternative way to close the S2R gap is by iterative data collection [5]–[8]. The common problem of these approaches is the necessity to execute untrained policies directly on the robot, which may raise safety concerns.

Domain randomization (DR) [9] is a simple albeit promising domain adaptation technique that is well suited for S2R. It compensates for the discrepancy between different domains by extensively randomizing parameters of the training domain in simulation. DR has been successfully applied for transferring visual features [9] and high level policies [10].

The approach most related to ours is the work of Hwangbo *et al.* [11]. The authors demonstrate transferring of a low-level stabilizing policy for the Hummingbird quadrotor that is trained in simulation. In contrast to their work i) we assume minimal prior knowledge about quadrotor's dynamics parameters, ii) we transfer a single policy to multiple quadrotor platforms, iii) we do not use auxiliary pre-tuned PD controller in the learned policy, iv) we simplify the cost function used for training the policy, v) we investigate the importance of different model parameters and the role of DR for S2R transfer of quadrotor's low-level policies.

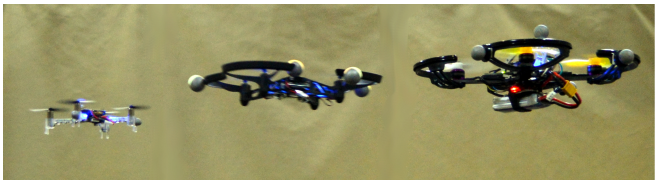

Fig. 1. Three quadrotors of different sizes controlled by the same policy trained entirely in simulation.

## II. Problem Statement

The quadrotor state is described by the tuple $(\mathbf{e}_p, \mathbf{e}_v, R, \mathbf{e}_\omega)$, where $\mathbf{e}_p \in \mathbb{R}^3$ is the position error, $\mathbf{e}_v \in \mathbb{R}^3$ is the linear velocity error in the world frame, $R \in SO(3)$ is the rotation matrix from the quadrotor's body coordinate frame to the world frame, and $\mathbf{e}_\omega$ is the angular velocity error in the body frame. The objective is to minimize the norms of $\mathbf{e}_p, \mathbf{e}_v, \mathbf{e}_\omega$ and drive the last column of $R$ to $[0,0,1]^T$ in the shortest time. The policy should be capable of recovering from different initial conditions, as well as transferable to other quadrotor platforms while retaining high performance.

## III. Dynamics Simulation & Learning

We simulate the dynamics using Newton-Euler equations. We simulate sensor noise, and non-ideal motors with motor lag and motor noise to increase realism. Motor lag is simulated by a discrete-time first-order low-pass filter. Motor noise is added following a discretized Ornstein-Uhlenbeck process.

We use a fully-connected neural network to represent a policy. The network input is an 18-dimensional vector representing the quadrotor state presented in Section II. This state representation enables trajectory tracking by shifting the goal state. The output of the network is the normalized commands $\mathbf{a}$ for individual rotors. The policy is trained using the Proximal Policy Optimization (PPO) algorithm [12]. During training, we sample the initial states uniformly. The goal state is always selected to hover at $[0,0,2]^T$ in the world coordinates. The cost function is defined as

$$c_t = (\|\mathbf{e}_p\|_2 + \alpha_v \|\mathbf{e}_v\|_2 + \alpha_\omega \|\mathbf{e}_\omega\|_2 + \alpha_a \|\mathbf{a}\|_2 + \alpha_R \cos^{-1}((\mathrm{Tr}(R)-1)/2))dt, \quad (1)$$

This paper is an extended abstract of work submitted to IROS 2019 (see https://arxiv.org/abs/1903.04628). All authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA. Star (*) refers to equal contribution. Email: {molchano, taochen, whoenig, japreiss, ayanian, gaurav}@usc.edu

TABLE I

ROBOT PROPERTIES.

| Robot | CF | Small | Medium |
|---|---|---|---|
| Weight [g] | 33 | 73 | 124 |
| $l_{body;w}$ [mm] | 65 | 85 | 90 |
| $r_{t2w}$ (approximate) | 1.9 | 2.0 | 2.7 |

where $\alpha_\omega, \alpha_a, \alpha_R, \alpha_v$ are non-negative scalar weights. The term $\cos^{-1}\left((\mathrm{Tr}(R)-1)/2\right)$ represents the angle of rotation between the current orientation and the identity rotation matrix.

We investigate the role of DR for generalization toward models with unknown dynamics parameters. During training, we sample dynamics parameters for each individual trajectory. We experiment with two approaches for dynamics sampling (parameter randomization):

1) Randomization around a set of nominal values.
2) Randomization within a set of limits (full randomization).

## IV. EXPERIMENTS

We analyze the influence of the different terms in the cost function (1) on the flight performance of the Crazyflie 2.0 platform. We measure the performance in terms of mean position error $e_h$, mean angular error $\bar{\mathbf{e}}_\theta$, and oscillation frequency $f_o$. We show that we can train a successful policy with a cost that only penalizes position, angular velocity, and actions. The major advantage of adding the cost on rotational errors is yaw stabilization, which might be desired for takeoff or if the quadrotor is carrying a camera.

We test the influence of noise and motor lag (settling time) in a trajectory tracking task on the Crazyflie 2.0. The task includes flying a figure-eight at moderate speeds (up to 1.6 m/s, 5.4 m/s$^2$, 24 deg roll angle; 5.5 s long), and landing. As a baseline, we use the non-linear controller ("Mellinger") that is part of the Crazyswarm [13] with default gains and zero integral terms. Our neural network with the motor settling time $T = 0.15$ has a mean position error of 0.19 m, which is similar to the Mellinger controller (0.2 m). A network trained without motor lags ($T$ nearly zero) overshoots frequently and has a larger mean position error of 0.21 m. If the network is trained without sensor and motor noise, we measure a mean position error of 0.24 m. Note that none of our policies are explicitly trained for trajectory tracking. Nonetheless, they still show competitive tracking performance compared to the baseline trajectory-tracking controller specifically tuned for the Crazyflie.

We investigate how well a single policy works across three different quadrotors with varying physical properties: Crazyflie 2.0, small, and medium size as described in Table I and shown in Fig. 1. We found that our policies showed comparable performance to the Mellinger controllers on all platforms, and full randomization shows more consistent results over all platforms, but for specific platforms may perform worse than other policies.

We perform recovery robustness tests by making repetitive throws of the quadrotors in the air. Our policies perform especially well on the Crazyflie platform recovering from 80% of all throws and up to 100% throws with moderate attitude changes ($\leq 35°$)[1]. The policy also shows substantial level of robustness on other quadrotors. Another surprising observation is that the control policies can deal with much higher initial velocities than those encountered in training ($\leq$ 1 m/s). Fig. 2 shows an example of a recovery trajectory.


Fig. 2. An example of a recovery trajectory from a random throw with an initial linear velocity of approximately 4 m/s.

## V. FUTURE WORK

Our findings open exciting directions for future work. We are planning to explore how we can incorporate a limited number of samples collected from real systems with our policies to improve the trajectory tracking performance of our policy without manual tuning. We are also planning to investigate if different network structures are able to transfer better to multiple platforms. Finally, we want to explore if we can increase robustness of the policies by training with different failure cases such as broken motors.

## REFERENCES

[1] K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov, "Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system," in *SIMPAR*, 2018.

[2] J. Tan, T. Zhang, E. Coumans, *et al.*, "Sim-to-real: Learning agile locomotion for quadruped robots," in *RSS*, 2018.

[3] R. Antonova, S. Cruciani, C. Smith, and D. Kragic, "Reinforcement learning for pivoting task," *CoRR*, 2017.

[4] J. Förster, "System identification of the crazyflie 2.0 nano quadrocopter," BA Thesis, ETH Zurich, 2015.

[5] P. F. Christiano, Z. Shah, I. Mordatch, *et al.*, "Transfer from simulation to real world through learning deep inverse dynamics model," *CoRR*, 2016.

[6] Y. Chebotar, A. Handa, V. Makoviychuk, *et al.*, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," *CoRR*, 2018.

[7] S. Zhu, A. Kimmel, K. E. Bekris, and A. Boularias, "Fast model identification via physics engines for data-efficient policy search," in *IJCAI*, 2018.

[8] J. Tan, Z. Xie, B. Boots, and C. K. Liu, "Simulation-based design of dynamic controllers for humanoid balancing," in *IROS*, 2016.

[9] J. Tobin, R. Fong, A. Ray, *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IROS*, 2017.

[10] J. Tremblay, T. To, A. Molchanov, *et al.*, "Synthetically trained neural networks for learning human-readable plans from real-world demonstrations," in *ICRA*, 2018.

[11] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *RA-L*, 2017.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017.

[13] J. A. Preiss*, W. Hönig*, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *ICRA*, 2017.

[1]Computed as a 95-th percentile in our experiments.