AQLUUC: Automatic Quadcopter Launching Under Unstructured Conditions

Ian Park, Michael Chambers, Nathaniel Chan

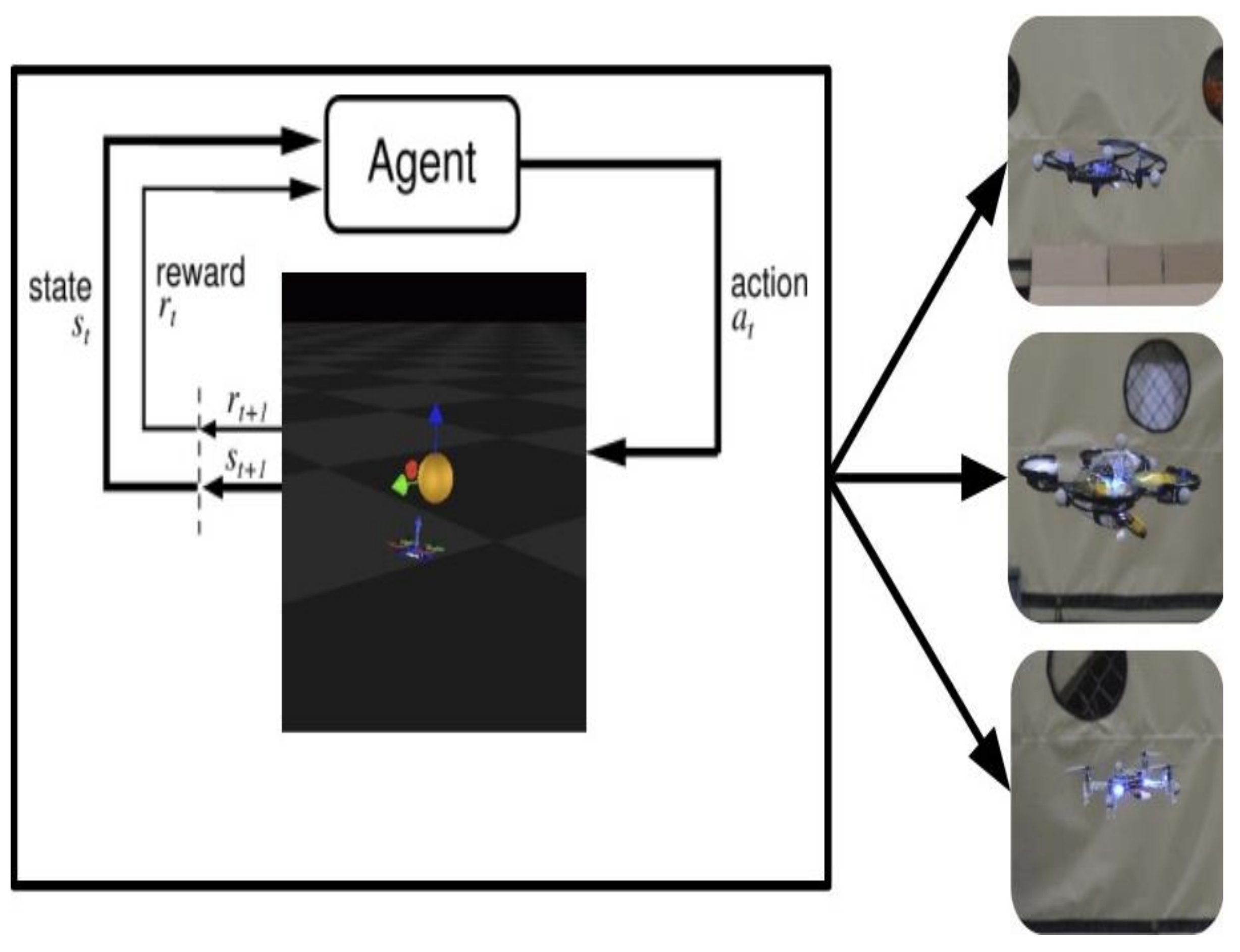
INTRODUCTION

The flexible launching of a quadcopter system remains challenging due to its highly dynamical nature. Generally, the popular commercial quadcopter, such as DJI, requires a trained operator to execute the launching on the plain ground with remote control. Such requirements cannot be always satisfied under certain extreme environments or emergencies.

In this project, the objective is to design an automatic launching module using reinforcement learning (RL) for a nano-quadcopter system, i.e., Crazyflie 2.1, which supports more flexible launching under unstructured conditions, such as rugged ground and hand-throwing.

To achieve the best possible launching/flight performance, we bring together a number of different techniques and experiments, such as (1) simulation-to-reality transfer learning, (2) robust adversarial reinforcement learning algorithm, (3) domain randomization, and (4) the selection of a good reward function.

TRANSFER LEARNING



- Use reinforcement learning to train policies in simulation that transfer remarkably well to multiple different real-world quadcopters.
- Train without any pre-tuned stabilizing controllers and directly control the motor thrusts.
- Direct control is based on a control policy, which is an auto-generated *C*-function from the trained neural network model with simulated trajectories.

ROBUST ADVERSARIAL REINFORCEMENT LEARNING (RARL)

Robust Adversarial Reinforcement Learning (RARL)

```
1   Input: Environment $\mathcal{E}$; Stochastic policies $\mu$ and $\nu$
2   Initialize: Learnable parameters $\theta_0^\mu$ for $\mu$ and $\theta_0^\nu$ for $\nu$
3   for $i = 1,2,..N_{iter}$ do
4       $\theta_i^\mu \leftarrow \theta_{i-1}^\mu$
5       for $j = 1,2,..N_\mu$ do
6           $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \mathrm{roll}(\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_{i-1}^\nu}, N_{traj})$
7           $\theta_i^\mu \leftarrow \mathrm{policyOptimizer}(\{(s_t^i, a_t^{1i}, r_t^{1i})\}, \mu, \theta_i^\mu)$
8       end for
9       $\theta_i^\nu \leftarrow \theta_{i-1}^\nu$
10      for $j = 1,2,..N_\nu$ do
11          $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \mathrm{roll}(\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_{i-1}^\nu}, N_{traj})$
12          $\theta_i^\nu \leftarrow \mathrm{policyOptimizer}(\{(s_t^i, a_t^{2i}, r_t^{2i})\}, \nu, \theta_i^\nu)$
13      end for
14  end for
15  Return: $\theta_{N_{iter}}^\mu, \theta_{N_{iter}}^\nu$
```

1. RARL optimizes both agents, starting with the protagonist (line 4-8) and then the adversary (line 9-13).

1. Both agents' initial parameters are sampled from random distribution.

1. The *roll* function samples *N* trajectories given the environment definition, protagonist's policies, and adversary's policies.

1. Both agents' optimized parameters are computed to maximize the reward function.

1. Repeat until convergence.

DOMAIN RANDOMIZATION

Domain randomization is a technique that randomizes a training environment's dynamics to help the trained policies be more robust to model errors. This technique moderately improves the transferability of the model from simulation to hardware.

In each episode of training, we randomize parameters of the environment either around nominal values or within limits. Typically, parameters are randomized such that the environment closely resembles various real-world scenarios.

TABLE I
RANDOMIZATION VARIABLES AND THEIR DISTRIBUTIONS.

Variable	Unit	Nominal Randomization	Total Randomization
m	kg	0.028	≤ 5
$l_{body,w}$	m	0.065	$\sim \mathcal{U}(0.05, 0.2)$
T	s	0.15	$\sim \mathcal{U}(0.1, 0.2)$
r_{t2w}	kg/N	1.9	$\sim \mathcal{U}(1.8, 2.5)$
r_{t2t}	s^{-2}	0.006	$\sim \mathcal{U}(0.005, 0.02)$

REWARD FUNCTION

We compare two types of rewards in training the quadcopter: the reward given the Sim-to-(Multi)-Real (S2R) paper and the time-to-reach-based (TTR-based) reward.

The TTR function combines the benefits of model-free RL and optimal control to alleviate the data inefficiency present in model-free RL. Since TTR is computationally intractable for high-dimensional systems, TTR trains using a subset of states given in the system model.

For the quadcopter, we train the TTR function on two subset of states:

$$\{V_x, V_z, \theta, W_t\} \quad (1)$$

$$\{V_y, V_z, \phi, W_p\} \quad (2)$$

DISCUSSION & FUTURE WORK

Our current work involves testing and comparing reward functions, randomized environmental parameters, and RARL. By the end, we hope to have trained a quadcopter model robust enough to launch under most unstructured launching conditions and be able to draw a conclusion on which training configuration leads to the best result.

As an extension of the current work, we would like to explore if we can further improve the trained policy for even more increased robustness. A continuation of our work would involve training the quadcopter to safely and automatically make an emergency landing on the ground, such as in the case when one or more motors and/or propellers fail.

REFERENCES

1. Molchanov, Artem, et al. "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors." *arXiv preprint arXiv:1903.04628* (2019).

1. Pinto, Lerrel, et al. "Robust adversarial reinforcement learning." *International Conference on Machine Learning*. PMLR, 2017.

1. Lyu, X., & Chen, M. "TTR-Based Reward for Reinforcement Learning with Implicit Model Priors." *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2020.