Time-optimal Flying a VTOL Drone

Larissa Rickler
larissa.rickler@tum.de

***Abstract*— We tackle the problem of minimum-time flight for a quadrotor through a sequence of waypoints while exploiting the quadrotor dynamics. We implemented a low-level neural network controller that uses deep reinforcement learning methods. The controller enables a drone to faithfully follow a path through a sequence of waypoints, while minimizing the required time for completion. We developed a completely learned policy, which measures inputs from its environment and performs all flight maneuvers without the need of an additional low-level PID controller.**

***Index Terms*—reinforcement learning, VTOL drone, quadrotor, low-level neural network controller**

I. Introduction

Vertical take-off and landing (VTOL) drone are extremely versatile. Possible application fields are logistics, infrastructure, search, rescue and many more [1]. Models, such as quadrotors, are highly agile and allow for rapid maneuvers [2], [3]. Hence, they are ideal for survivor search in mountains or after natural disasters like earthquakes, floods or wildfires. There is also a huge potential in autonomous flights [4].

Deep reinforcement learning methods have been successfully utilized to train a neural network controllers for minimum-time quadrotor flight [5]. This research adapted the results from Penicka et al. to an low-level controller that directly controls each rotor individually without the need of a underlying PID controller.

II. Methods

The approach we used consisted of 1) formulating a progress maximization along the trajectory while reaching the waypoints and 2) a curriculum training strategy to train a neural network policy using deep reinforcement learning. Furthermore, we initially reduced the problem setting to a 2D world with a different model, elektra VTOL3, before applying our findings in a 3D world and a quadrotor model. In the following, the policy architecture, reward function, learning strategy and training details that were used in order to train a minimum time flight policy is described.

A. Policy Architecture

The observations consist of two parts: the VTOL drone's state (body orientation, as well as the translational and rotational velocities) and the relative path to the next two waypoints. The observation vector is denoted as $o(t) = [R_W(t), v_B(t), \omega_B(t), wp_B(t), nwp_B(t)]$, where $R_W(t)$, $v_B(t)$, $\omega_B(t)$, are the drone's rotation matrix, velocity, body rates and $wp_B(t)$, $nwp_B(t)$ are the relative path to the next two waypoints of the trajectory. The action vector produced by the policy $a(t) = [f_1, f_2, f_3, f_4]$ gives the PWM signals for each rotor individually. In Figure 1 both observation and action space are illustrated, as well as the actor network architecture, which is a 2-layer multilayer perceptron (MLP). For more details on the network architecture see Section II-D.

B. Reward Function

The guiding trajectory is modeled as the connected lines between n waypoints $g_1, ..., g_n$. To calculate the progress at position p we define the closest point on the trajectory as $\psi(p)$ and its corresponding line segment index as $l(p)$ as

$$\begin{aligned} l(p), \psi(p) = \text{arg min}_{l(p),\psi(p)} \|p - \psi(p)\| \\ \text{s.t. } \psi(p) = g_{l(p)} + t(g_{l(p)+1} - g_{l(p)}), \\ t = \frac{(p - g_{l(p)}) \cdot (g_{l(p)+1} - g_{l(p)})}{\|g_{l(p)+1} - g_{l(p)}\|^2}, \\ l(p) \in \{1, .., n-1\}, t \in [0, 1]. \end{aligned} \tag{1}$$

The computation of the progress in time step t is illustrated in Figure 2.

The reward function consists of five main parts: encouraging progression along the trajectory in every time step, rewarding the overall reached distance along the trajectory, rewarding reached waypoints, penalizing high body rates ω and penalizing inactivity. This approach is leaned on the work of Penicka et al. [5]. The total reward function $r(t)$ at time t is defined as

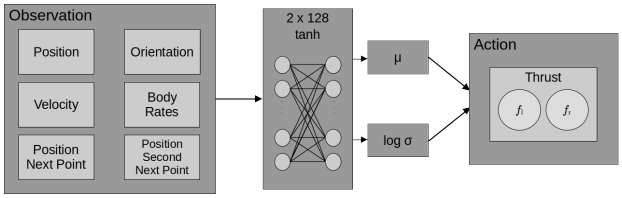
$$r(t) = k_p r_p(t) + k_s s(p(t)) + k_{wp} r_{wp} + k_\omega \|\omega\| - fall, \tag{2}$$

with the reached distance along the trajectory $s(p(t))$ and the progress reward at each time step $r_p(t)$ being defined as

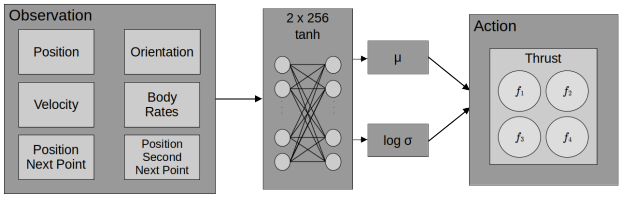
$$s(p) = \sum_{i=1}^{l(p)-1} \|g_{i+1} - g_i\| + \|\psi(p) - g_{l(p)}\|, \tag{3}$$

$$r_p(t) = s(p(t)) - s(p(t - \Delta t)). \tag{4}$$

The reached distance $s(p(t))$ is part of the reward function to counteract possible singularities due to sharp corners of the trajectory and the minimum distance projection. The reward r_{wp} was payed, if the drone reached a waypoint within a distance d_{wp} smaller than a tolerance r_{tol} for the first time. The term $k_\omega \|\omega\|$ discourages high body rates. The penalty term $fall$ was only subtracted if the drone fell under 0m due to inactivity. The contribution of each reward part is scaled by the hyperparameters k_p, k_s, k_{wp} and k_ω.



(a) Actor network for elektra VTOL3 controller in 2D world.



(b) Actor network for quadrotor controller in 3D world.

Fig. 1: Illustration of the actor networks with the used observation and action spaces.

C. Training Strategy

Optimizing the reward (2) directly could lead to suboptimal performance due to local minima and a higher risk of missing the waypoints in high speed flight. To overcome this we used a curriculum learning strategy.

First, in an initial slow learning phase a slow and stable policy is trained that follows the trajectory closely and stays within a velocity range between v_{min} and v_{max}. Second, a fast learning phase in which a racing policy was trained by encouraging velocities above a threshold v_{min} while still reaching the waypoints.

We achieve this by scaling the reward differently in the respective phase. The reward hyperparameters for progress k_p and distance k_s were scaled by a factor s_m that encouraged translation velocities between v_{min} and v_{max}. This scaling factor also motivates the drone to stay within a distance of d_{max} to the trajectory. Note that v_{min}, v_{max} and d_{max} differ in the different phases. In the fast learning phase a dummy value for v_{max} was inserted. The scaling factor s_m is calculated as

$$
\begin{aligned}
s_m &= s_v \cdot s_{gd}, \\
s_v &= \min(1, 10^{v_{max}-||v||}) \cdot \min(1, 10^{||v||-v_{min}}), \\
s_{gd} &= \min(1, e^{-\|p-\psi(p)\|+d_{max}}).
\end{aligned}
$$

Furthermore, to ensure that the drone still reaches the waypoints even in high speed flight k_p and k_s were scaled down in the fast learning phase such that the term for reaching a point $k_{wp} r_{wp}$ was in comparison more rewarding. Ablating this scale down would lead to uncontrolled speedup and therefore to missing waypoints. Moreover, the parameter k_ω was reduced in the fast learning phase to allow for more rapid maneuvering. See Table I for details on the parameters that were used for the respective phase.

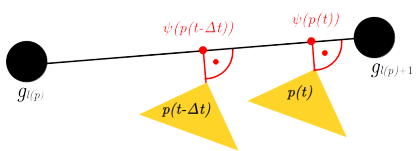


Fig. 2: Illustration of trajectory with two waypoints. The nearest point $\psi(p)$ on trajectory from drone at position p and its corresponding line segment index $l(p)$ is used to calculate the progress in each time step Δt and the reached distance.

D. Training Details

We used an actor-critic algorithm as approximator. The applied optimization algorithm is Adam. The actor network produces actions learned via the Proximal Policy Optimization (PPO). Both actor and critic are multilayer perceptrons (MLP), see Table II and Figure 1 for details on their architectures.

The policy was trained while utilizing 8 parallel agents. This increases the speed of collecting data and diversifies the experienced states and observations. For training we used the Julia environment Flyonic [6] that provides simulated models with the appropriate dynamics of elektra VTOL3 and a quadrotor.

The training trajectories consisted of n waypoints which were generated as follows:

$$
wp_1 = (0,0,0)^T, \; wp_i = wp_{i-1} + z_i \text{ for } i = 2, ..., n \qquad (5)
$$

where z_i was sampled from the multivariate uniform distribution $\mathcal{U}_3$ on $[-7,7] \times 0 \times [1.5,7] \subseteq \mathbb{R}^3$ for the 2D world setting and $[-3,3] \times [-3,3] \times [0.5,3] \subseteq \mathbb{R}^3$ for the 3D world setting. The distance ranges were decreased in the 3D setting since the quadrotor is significantly smaller than the elektra model.

III. Results and Discussion

To validate the trained policy, we tested it on 200 trajectories that were generated similarly as the training paths. We investigated the achieved success rate on reaching every point within the tolerance and the average velocity of the successful flights per training step for both the slow and fast learning phase. The fast flight policy was retrained with the model parameters after 0.5×10^6 of slow phase training. In [7] videos of the results of each of the following experiments can be found.

A. Evaluation of 2D World Setting

The success rate in the slow learning phase increases in a fast pace. After 500,000 steps the rate already consistently reaches values over 90%, as seen in Figure 3a. The average velocity of a successful slow flight approached $3\frac{\text{m}}{\text{s}}$ which corresponds to v_{max}, see Figure 3b.

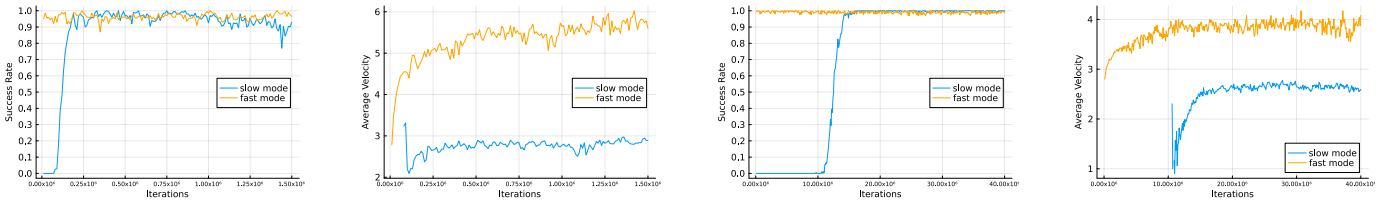
Figure 3a also shows that the success rate in the fast learning phase remains in almost ever step by over 90%, as . After 1.5×10^6 steps of the fast learning phase, the simulated elektra VTOL3 drone reaches speeds of up to $6\frac{\text{m}}{\text{s}}$ which is optimal for this model as illustrated in Figure 3b.

TABLE I: Parameters of the algorithm.

General parameters				Slow learning phase				Fast learning phase			
Variable	Value	Variable	Value	Variable	Value	Variable	Value	Variable	Value	Variable	Value
Δt [s]	0.025	$fall$ [-]	1	v_{min} [m/s]	1.0	k_p [-]	$5.0 \cdot s_m$	v_{min} [m/s]	4.0	k_p [-]	$\frac{5.0 \cdot s_m}{\sum_{i=1}^{n-1} \|g_{i+1} - g_i\|}$
n [-]	4	k_{wp} [-]	50.0	v_{max} [m/s]	3.0	k_ω [-]	0.01	v_{max} [m/s]	50.0	k_ω [-]	0.001
r_{tol} [m]	0.5	r_{wp} [-]	$\exp(\frac{d_{wp}}{r_{tol}})$	d_{max} [m]	0.5	k_s [-]	$\frac{2 v_{max} \Delta t \cdot s_m}{\sum_{i=1}^{n-1} \|g_{i+1} - g_i\|}$	d_{max} [m]	1.0	k_s [-]	$\frac{2 v_{max} \Delta t \cdot s_m}{\left(\sum_{i=1}^{n-1} \|g_{i+1} - g_i\|\right)^2}$

TABLE II: Network architectures for elektra VTOL3 controller in 2D world and for quadrotor controller in 3D world.

Dimension	Network	Type	Input	Hidden Layer	Activation	Last Layer	Output	Output Size
2D	actor	MLP	observation	2x128	tanh	1x128	action mean	2
						1x128	action variance	2
	critic	MLP	observation	2x128	tanh	1x128	value	1
3D	actor	MLP	observation	2x256	tanh	1x256	action mean	4
						1x256	action variance	4
	critic	MLP	observation	2x256	tanh	1x128	value	1



(a) Success rate per step for both learning phases in 2D.

(b) Average velocity per step for both learning phases in 2D.

(c) Success rate per step for both learning phases in 3D.

(d) Average velocity per step for both learning phases in 3D.

Fig. 3: Evaluation of our policy per training step. Discussed in Section III.

B. Evaluation of 3D World Setting

In the slow learning phase, the quadrotor model's success rate consistently reaches values between 96% and 100% after 15.0×10^6 steps as shown in Figure 3c. This is probably due to its high agility. Figure 3d shows that the average velocity again approaches $3\frac{\text{m}}{\text{s}}$ as encouraged by v_{max}.

The high success rate remains over 96% in the fast learning phase as illustrated in Figure 3c. After 40.0×10^6 steps the quadrotor's average speed seems to converge to $4\frac{\text{m}}{\text{s}}$ which was the value we chose for v_{min} as illustrated in Figure 3d. This is not time optimal yet. We hypothesize that the curriculum learning strategy needs to be extended to more phases in which v_{min} is slowly increased to achieve even better results.

C. Evaluation on the Office Trajectory

To compare our results to those of [5], we retrained our fast flight model on horizontal trajectories for another 40.0×10^6 steps. The trajectories were generated as in (5) with $z_i \sim \mathcal{U}_3([-8.0, 0.0] \times [-4.0, 4.0] \times 0)$. We than generated a collision free trajectory to mimic the path optimal trajectories through the so called office environment. The office is one of the test environments used in [5]. In [8] the environment and the generated trajectory is shown. Here we evaluated 30 runs. The success rate was 100% even though the test trajectory had much sharper curves than the training trajectories. The average velocity during those runs were $7\frac{\text{m}}{\text{s}}$. In Table III our results are compared with those of Penicka et al. The better results of their algorithm are due to an additional use of a path planning algorithm as well as their use of an low-level PID controller to convert their action signals to speed commands. This action modality has been identified as optimal for learned control policies for quadrotor by [9].

TABLE III: Comparison of success rate, average completion time and best completion time in the office environment.

	success[%]	avg. compl. time [s]	best. compl. time [s]
[5]	100	1.74	1.72
ours	100	3.61	3.55

IV. Conclusion

This paper adapted the findings of Penicka et al. [5] to a low-level neural network controller for time-optimal flight through a sequence of waypoints while utilizing deep reinforcement learning methods. The controller achieves 100% success rate on our test environment. However, our approach could not outperform their result on the test environment. This is due to a missing path planing algorithm in this work as well as the low-level controller approach which is known to be suboptimal [9]. Nonetheless, neural network low-level controllers might prove useful for more complicated air vehicles. Possible further research topics might be to extend the curriculum learning strategy to more phases in which the velocity is further increased slowly, retraining with domain randomization to achieve robustness to varying model parameters or using a physical model with latency.

REFERENCES

[1] J. Hilf and K. Umbach, "The commercial use of drones," *Computer Law Review International*, vol. 16, no. 3, pp. 65–71, Mar. 2015.

[2] E. Ackerman, "Ai-powered drone learns extreme acrobatics," *IEEE Spectrum*, Oct. 2020. [Online]. Available: https://spectrum.ieee.org/ai-powered-drone-extreme-acrobatics

[3] J. Verbeke and J. D. Schutter, "Experimental maneuverability and agility quantification for rotary unmanned aerial vehicle," *International Journal of Micro Air Vehicles*, vol. 10, no. 1, pp. 3–11, Oct. 2017.

[4] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, Oct. 2021.

[5] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, Jul. 2022.

[6] F. Süberkrüb, "Flyonic." [Online]. Available: http://docu.flyonic.de

[7] L. Rickler, "Advanced deep learning for robotic ws22/23." [Online]. Available: https://www.youtube.com/playlist?list=PLfNlYts0db6vorCQ8P05cJyFHPT-STbDf

[8] L. Rickler, "Advanced deep learning for robotic ws22/23." [Online]. Available: https://youtu.be/TuyNiLsGyxo

[9] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," pp. 10 504–10 510, 2022.