Pixel-based control for Fixed-wing UAVs
CS5180 Fall 2022

Tony Smoragiewicz

Northeastern University
360 Huntington Ave., Boston, MA 02115
smoragiewicz.t@northeastern.edu

Project Description

Pixel-based control has recently become an interesting research topic for robotics. Using pixels is tempting because vision is biologically inspired and because quality camera sensors have become very inexpensive. A modern approach to this was done by (Levine et al. 2015) where reinforcement learning and optimal control was combined with convolutional neural networks (CNNs) to find control policies for a robot by interacting with the environment. Since then, more and more research has been conducted to achieve fast, accurate results on dynamical systems. Very recent results from (Bleher, Heim, and Trimpe 2022) have shown progress towards standardizing the process on unstable dynamical systems (Furuta Pendulum).

I plan to continue this trend by finding control policies for dynamic systems based entirely from camera pixels. My work will be with a fixed-wing drone with only longitudinal dynamics being considered. Constraining the simulation to 2D will greatly simplify the dynamics. This will allow much more time and effort to be dedicated to solving the reinforcement algorithms. Recent successes have also been achieved in the aerial robot domain by (Imanberdiyev et al. 2016a), (Bøhn et al. 2019), and (Imanberdiyev et al. 2016b). With the later even being applied to navigation. I was surprised by the many recent successes. They shall be invaluable as I try to replicate and extend their achievements.

MDP Formulation

The agent will be unaware of the underlying state equations needed to model the aerodynamics. Instead, the agent will receive a vector of pixels with three channels corresponding to hue, saturation, and value from a front facing camera. This is a common color space in computer vision. The representation of the state $\mathcal{S}$ can be seen below:

$$H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} S = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

.

The agent has two actions that can be applied at every time step. These are engine thrust (T) and elevator deflection (δ_e). Representation of the actions $\mathcal{A}$ can be seen below:

$$0 \leq T \leq T_{max}$$
$$-\delta_{e_{max}} \leq \delta_e \leq \delta_{e_{max}}$$

The agent will receive a reward $\mathcal{R}$ of +1 on every time step while not in the terminal state. In addition to this positive reward, two negative rewards are represented as engine thrust and absolute change in elevator deflection. The coefficients k_1 and k_2 are there to scale the rewards to a reasonable level. This is done because changing the elevator deflection takes much less energy than turning a propeller.

$$-k_1 \times \frac{T_t}{T_{max}}$$
$$-k_2 \times \frac{|\delta_{e_t} - \delta_{e_{t-1}}|}{\delta_{e_{max}}}$$

I am unsure about discounting and episodic vs continuing at the moment. This will require more thought on how to frame this problem. A sample visualization of the proposed simulator can be seen in Figure 1.

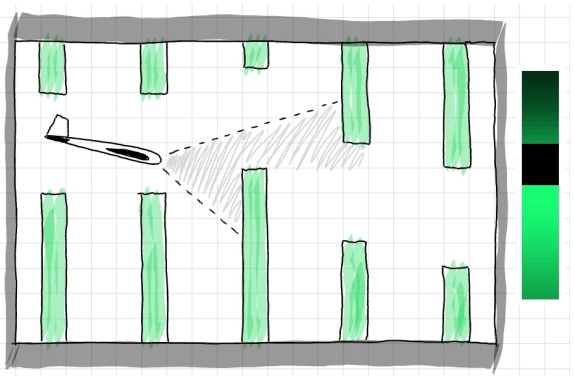

Figure 1: Sample drawing of the simulation. The grey slice represents the field-of-view of the front-facing camera. The vertical bar on the right side is the projection from the UAV's camera view. Note that the green obstacles are further apart in reality. In addition, the bar on the right is actually a vector and not a matrix.

Simulator

A simulator does not exist for this particular environment but I have already started to write the Python code. My undergraduate studies were in Aerospace Engineering so I feel comfortable writing the underlying dynamical equations to simulate the fixed-wing UAV in longitudinal flight.

The coefficients of lift, drag, and pitching moment:

$$\begin{aligned} C_L =& C_{L0} + C_{L\alpha}\alpha \\ C_D =& C_{D0} + KC_L^2 \\ C_{\mathcal{M}} =& C_{\mathcal{M}0} + C_{\mathcal{M}\alpha}\alpha + C_{\mathcal{M}\dot{\alpha}}\dot{\alpha} + C_{\mathcal{M}\delta_e}\delta_e \end{aligned}$$

The lift force, drag force, and pitching moment:

$$\begin{aligned} L =& \frac{1}{2}\rho V^2 S C_L \\ D =& \frac{1}{2}\rho V^2 S C_D \\ \mathcal{M} =& \frac{1}{2}\rho V^2 S c C_{\mathcal{M}} \end{aligned}$$

The pitch angle:

$$\theta = \gamma + \alpha$$

The underlying nonlinear time-invariant dynamical equations used for numerical integration:

$$\begin{aligned} m\dot{V} =& -D - mg\sin\gamma + T\cos\alpha \\ -mV\dot{\gamma} =& -L + mg\cos\gamma + T\sin\alpha \\ \dot{x} =& V\cos\gamma \\ \dot{z} =& -V\sin\gamma \\ \ddot{\theta} I_{yy} =& \mathcal{M} \end{aligned}$$

Algorithms

I plan to use Proximal Policy Optimization (PPO) as the baseline algorithm. I am choosing PPO because (Azar et al. 2021) gave an in depth analysis of varying deep RL algorithms with PPO scoring first or second on a wide range of drone-related control tasks. This trend was confirmed by (Bøhn et al. 2019) as well. If I am able to find moderate success, I will move to a Recurrent PPO / PPO with Long, Short-Term Memory (LSTM). This seems like a natural progression because I am working with time-series data. In my advanced computer vision class, we learned how Recurrent CNNs can achieve more accurate results in dynamic environments. Because the state space is limited to a vector and not a real 2D image, a suspect that training time will be significantly reduced.

Results

I expect to find a control policy that successfully guides the UAV through the "obstacle course". Flight controllers are helpful because they stabilize an aircraft without human input or at least little human input. Because of this, I will compare results of the policy to the results of a person flying through the simulator. With anything there is a risk of failure but I will reduce this risk by making consistent progress throughout the next month and a half of the semester. In the event that I don't finish, I will still share my results and emphasize where I did find success in addition to what will be modified as I continue working. Most research papers aren't humble and fail to show what didn't work well from the start. It's helpful for others who follow in my steps to avoid these pitfalls early on.

References

Azar, A. T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H. A.; Ibrahim, Z. F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A. M.; Hameed, I. A.; and Casalino, G. 2021. Drone Deep Reinforcement Learning: A Review. *Electronics*, 10(9).

Bleher, S.; Heim, S.; and Trimpe, S. 2022. Learning Fast and Precise Pixel-to-Torque Control: A Platform for Reproducible Research of Learning on Hardware. *IEEE Robotics Automation Magazine*, 29(2): 75–84.

Bøhn, E.; Coates, E. M.; Moe, S.; and Johansen, T. A. 2019. Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization. *CoRR*, abs/1911.05478.

Imanberdiyev, N.; Fu, C.; Kayacan, E.; and Chen, I.-M. 2016a. Autonomous navigation of UAV by using real-time model-based reinforcement learning. In *ICARCV*, 1–6. IEEE. ISBN 978-1-5090-3549-6.

Imanberdiyev, N.; Fu, C.; Kayacan, E.; and Chen, I.-M. 2016b. Autonomous navigation of UAV by using real-time model-based reinforcement learning. In *ICARCV*, 1–6. IEEE. ISBN 978-1-5090-3549-6.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2015. End-to-End Training of Deep Visuomotor Policies. Cite arxiv:1504.00702.