Visual Servoing for Fixed-wing UAVs

Tony E. Smoragiewicz

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

Abstract

Visual servoing (VS) has gained interest in recent years. Much of this interest is due to the reduced cost of cameras, increased computation capabilities, and more efficient learning algorithms. In this worked, I investigated the use of Proximal Policy Optimization (PPO) algorithms as a method to find a control policy from a one-dimension camera feed for a fixed-wing drone in longitudinal flight. I observed that the policy from PPO was able to successfully guide the drone through a simulated environment with only having access to the camera stream. Because the drone is able to achieve a its long-term goal of completing the environment and short-term goal of using minimum control input, the policy effectively combines planning and control into a single process.

Introduction

The vast majority of aircraft flight around the world occurs in an obstacle free environment at high altitude. Because of this, planning and control are separate problems often run at rates of multiple orders of magnitude difference. The planning takes care of the route of the aircraft and the control takes care of correcting deviations from that route. This type of hierarchical navigation is possible because the obstacle free path and linearized dynamical equations around stable points. Aerospace technology and governmental regulations have changed over the last decade to make flight inside crowded airspace a realistic option for the near future. This market is expected to expand as autonomous drone deliveries become widespread as technology continues to mature.

Reliable control of fixed-wing aircraft has been possible for a few decades with the help of PID and state-feedback control. In contrast to modern control techniques that use vehicle states such as pitch, roll, yaw, (and their respective derivatives) for feedback control, new control techniques seek to use feedback that describes the state of the world as well. Cameras are one such way of providing information on the state of the world. Cameras provide much more data than magnetic, angular-rate, and gravity (MARG) sensors, but much less ”information” without additional processing. Recent improvements in neural networks have provided engineers with an additional tool to extract meaningful information from this additional data. A promising type of deep learning algorithm in the field of reinforcement learning is Proximal Policy Optimization (PPO) introduced in 2017 by Open AI (Schulman et al. 2017).

My experiments show that PPO-Clip has the potential to control a fixed-wing drone in longitudinal flight while avoiding obstacles in a simulated environment. I compare varying approaches and versions in my results.

Background

Any background information needed to understand the methods used in the project (e.g., a description of a general search problem or some simpler algorithms that you build off of).

Markov Decision Processes (MDP) are often used in Reinforcement Learning to describe the interaction of an agent with the environment. Is this model of the world, an agent interacts and learns from the environment by taking an action and observing a reward and resulting state. The states often describe how the agent is positioned or located in the world but they could also include information about the internal state like happiness of a person or battery voltage of an electric car. MDPs contain all necessary information to separate future states from past states, given a current action. The next state is formulated with a transition function that's often stochastic. Although we like to think of the world as fully observable in small toy problems, a more realistic world is where the state is partially observable. Unsurprisingly, these class of problems are called Partially Observable Markov Decision Processes (POMDP). This type of formulation will be used later to describe the problem at hand.

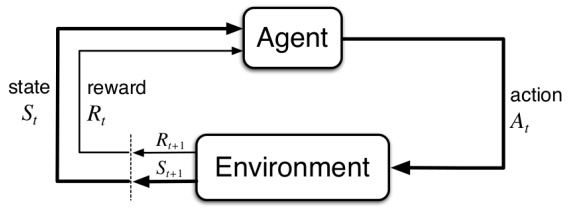

Figure 1: In general, the environment represents anything that cannot be changed by the agent. Oftentimes this can be part of the agent itself. Agent-Environment Interaction from (Sutton and Barto 2018)

Robotics is so broad it represents the combination of nearly every field of engineering. Within robotics, two major fields are localization and control. These tasks are often intertwined out of necessity. It's hard to know what actions to take if you don't know where you are, and on the other hand, it's hard to know where you are unless you know what actions you've taken. Localization can be thought of as the process to determine the state of the agent and control can be thought of as determining the best action to take. There are many approaches to solve both of these problems. Recently, PPO algorithms have shown state-of-the-art results on a wide variety of control tasks.

Related work

In the field of aerospace engineering, the career title for solving the problem at hand is called Guidance, Navigation, and Control. Guidance is often synonymous with Planning in robotics. Historically, it's standard practice to solve these problems separately. Planning is carried out over a longer horizon than control so this loop usually takes longed to compute but doesn't need to be calculated as often as the control loop. When the horizon lacks many obstacles, this approach is best but when the environment is very cluttered, the planning and control steps can merge closer together. This is the very reason that I have merge this problems into a single task for my research topic. Recent successes with neural networks have been achieved in the aerial robot domain by (Imanberdiyev et al. 2016a), (Bøhn et al. 2019), and (Imanberdiyev et al. 2016b). Although most problems only address control, the later was even applied to navigation.

Project description

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 corresponding to the view from a front facing camera. The representation of the state $\mathcal{S}$ can be seen below:

$$S = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

.

The agent has discrete action space of 30 actions, one of which that can be applied at each time step. These actions are the difference angles of the elevator deflection (δ_e). Representation of the actions $\mathcal{A}$ can be seen below:

$$-\delta_{e_{max}} \leq \delta_e \leq \delta_{e_{max}}$$

The agent will receive a reward $\mathcal{R}$ of +30 on every time step while not in the terminal state. In addition to this positive reward, a negative reward is absolute change in elevator deflection. The reward of +30 was chosen because this forces $\mathcal{R} \geq 0$ at each timestep because there is a range of ± 15 degrees in elevator angle.

$$\frac{|\delta_{e_t} - \delta_{e_{t-1}}|}{\delta_{e_{max}}}$$

I used a discount factor of 0.99 and episodic tasks instead of continuing because much of the simulated environment was repetitive.

A sample visualization of the proposed simulator can be seen in Figure 10.

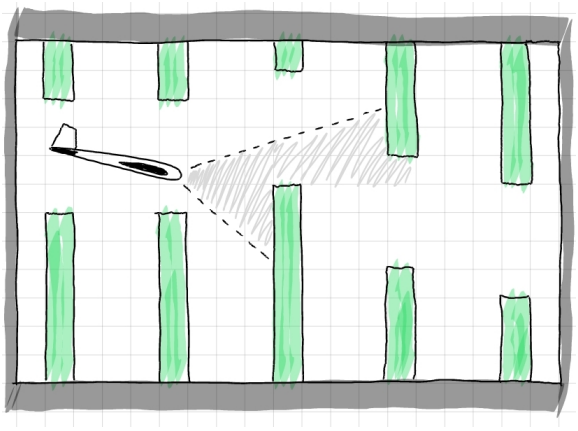

Figure 2: Sample drawing of the simulation. The grey slice represents the field-of-view of the front-facing camera. The vertical bar on the right side is the projection from the UAV's camera view. Note that the green obstacles are further apart in reality.

Algorithm

Pseudo-code for Proximal Policy Optimization can be seen in the figure below. PPO has similar reliability to trust region methods but easier implementation.

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

Figure 3: Clip version of Proximal Policy Optimization algorithm from Open AI.

Simulator

A simulator did not exist for this particular environment so I created one that merged with the standard Gym environment from Open AI. This approach was done because there are already many algorithms that use the next state, reward, and terminated outputs to train an RL agent.

The dynamical equations were linearized around a zero degree angle of attach and prospective project equations

were used to map the obstacles in the world onto the camera sensor in the simulator. The depth Z, height X, and focal length f are used in the equation below. The dynamics and camera view represent all information needed to create the agent and environment for a POMDP.

$$x = f\frac{X}{Z}$$

The vector of pixels was given a larger height in the figures below purely for visualization purposes.

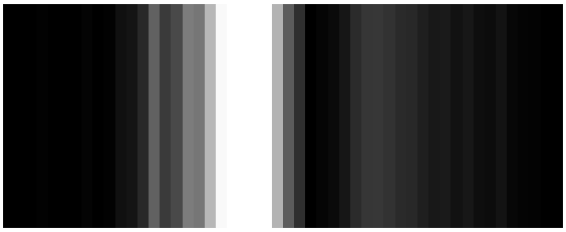

Figure 4: Sample image from the drone's camera feed.

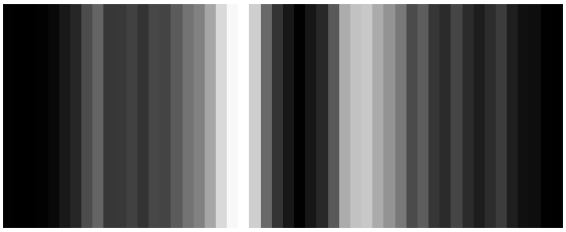

Figure 5: Sample image from the drone's camera feed.

Experiments

The first step in experimentation was verifying that PPO was working correctly. The Open AI gym Cartpole task was used to confirm this. Training results are be seen below. After a short amount of time, a near optimal polciy was found.

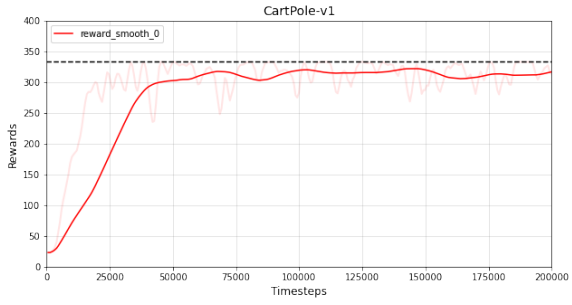

Figure 6: Training on CartPole

Next I performed a uniform hyperparameter search over a realistic range of values from similar studies. It was clear that very low clipping ratios were the only thing that made training much worse.

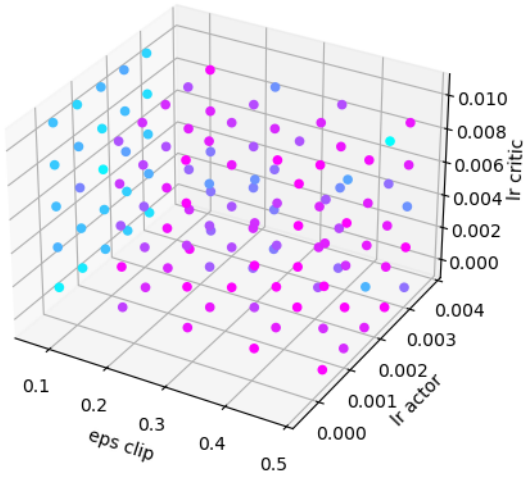

Figure 7: Hyperparameter Search

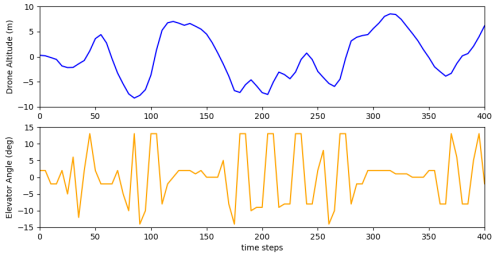

Figure 8: Results after little training

Conclusion

A summary of the results and code can be found at Github Repository

References

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.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

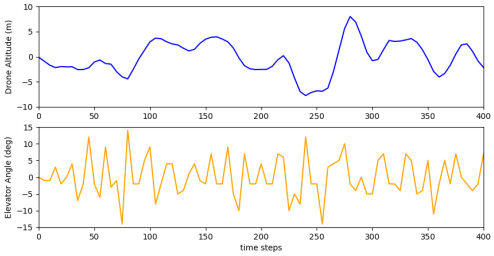

Figure 9: Results are more training

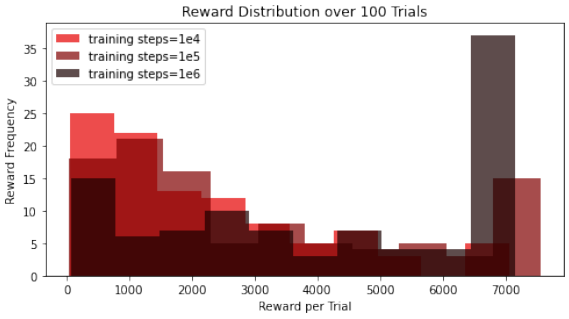

Figure 10: Testing