CS394R Project Proposal

Alejandro Moreno, Arnav Iyer

March 2022

1 Introduction

Drones are increasingly relevant in the defense and delivery spaces, so we thought it would be interesting to apply RL to this space. In particular, tuning PID controllers to balance attitude (pitch, roll, and yaw) is one area which is conducive to RL. Even though there are existing methods that accomplish the same goal, many of them approximate the non-linearity of the drone's actuators as linear systems with some error, and use multiple PID controllers for different drone speeds to cope with this. Using RL could allow for online PID adjustment for different conditions.

2 Methodology

To accomplish some of our larger goals, we have broken down our process into a series of steps that increase in scope. Each step has a brief outline of the states, actions, and rewards we plan to use. θ, ϕ, and ψ represent the attitude of the drone, and their dotted counter parts represent their rate of change. v_x,v_y, and v_z represent the velocities of the drone.

First step: auto-tune an attitude controller (PID or LQR) using a simplified model of the drone dynamics. We propose to use just one just one axis (e.g. roll) and make sure that the controller stabilizes it. This is much simpler because there are no coupled dynamics between different axis.

- states(ϕ,$\dot{\phi}$)
- actions(k_p,k_i,k_p)
- reward $J = x^T S x + \int_{t_0}^{t_f} [x^T Q x + u^T R u] dt$

Second step: We can auto-tune multiple PIDs for each axis of the drone (e.g. pitch, yaw and roll). This is more challenging because of coupled dynamics; changes in one of pitch, yaw, roll will affect others.

- states(θ,ϕ,ψ,$\dot{\theta}$,$\dot{\phi}$,$\dot{\psi}$)
- actions(k_p,k_i,k_p) for each axis

- reward $J = x^T Sx + \int_{t_0}^{t_f} [x^T Qx + u^T Ru]dt$

Third step: We can also explore different velocity regimes which most certainly will change the dynamics of the drone. This could lead to an online PID tuning for a moving drone.

- states(v_x,v_y,v_z, θ,ϕ,ψ, $\dot{theta}$,$\dot{\phi}$,$\dot{\psi}$)
- actions(k_p,k_i,k_p) for each axis
- reward $J = x^T Sx + \int_{t_0}^{t_f} [x^T Qx + u^T Ru]dt$

Fourth step: We can move to a more detailed physics simulator like Unity to create nice visualizations and enable further development.

The reward function we want to use is based on the cost functional that is used in optimal control theory. It takes into account steady-state errors, errors in the state while evolving to a reference input and the magnitude of the action are also penalized (so that you don't overwhelm the actuators). In this sense, it seems to be a good reward function for our problem.

It is worth mentioning that not all stages need to be addressed. Just with the first one we would have a complete project that uses RL to control a simplified version of a drone. However, in case we make progress quickly we can get our hands into more realistic scenarios.

2.1 Environment

Our main source of data will be a drone dynamics simulator that Alejandro is developing for another class. We can use as an environment for training. If we get to the later stages of the project, we will consider using Unity to simulate a drone.

2.2 RL Techniques

Regarding RL techniques that we are thinking about using, Actor-Critic methods or QTopt may work for our purposes because we have continuous state spaces (angle of pitch, roll, yaw) as well as continuous action spaces (PID gains).

3 Open Questions

There are still a few things we need to think about regarding the project, we may want to explore different reward functions. We think that the best contender is to use ideas from control theory and penalizing a sum of the squared error of the PID controller, since this is what we want to minimize. However, we may want to see if other reward signals, like having a constant negative reward when the drone loses balance and zero otherwise.