Inverted Pendulum on a Quadcopter: A Reinforcement Learning Approach

Physical Sciences

Alexandre El Assad
aelassad@stanford.edu

Elise Fournier-Bidoz
efb@stanford.edu

Pierre Lachevre
lpierre@stanford.edu

Javier Sagastuy
jvrsgsty@stanford.edu

November 20th, 2017

CS229 - Project Milestone

1 Motivation

Balancing an inverted pendulum problem is a classic control problem. In our project we intend to implement an RL algorithm to stabilize drone flight and balance an inverted pendulum coupled to the body of the drone.

If time allows, we would like to test our learned policy using a simulator on a real flying quadcopter to show the effectiveness of our algorithm.

2 Method

Our project consists primarily of three parts: a physics simulator of the dynamics of the coupled quadcopter and inverted pendulum system, an RL algorithm and a reward function that the RL algorithm will attempt to maximize.

2.1 Simulating quadcopter dynamics

The Matlab simulator is based on the dynamics outlined in the paper by Hehn et.al. [1]. These dynamics couple the dynamics of a quadcopter with an inverted pendulum attached to its center of mass. The state vector X is $[x, y, z, \dot{x}, \dot{y}, \dot{z}, \alpha, \beta, \gamma, a, b, \dot{a}, \dot{b}] \in \mathbb{R}^{13}$ (position and velocity of the drone, Euler angles, position and velocity of the center of mass of the stick). The control vector U is $[a, w_x, w_y, w_z] \in \mathbb{R}^4$ (the mass normalized collective thrust, rotational rates about the vehicle body axis). From this model, a discrete time update function was developed.

2.2 Solving strategies

Two different learning methods were experimented with at the time of this milestone, tree search and value iteration. In both cases, the action space had to be discretized. For that purpose, we allowed 5 values for each of the dimensions of the control vector U for a total action space size of $5^4 = 625$, which is a reasonable size to compute. Furthermore, with a small enough time step, the drone will still have a large array of possible trajectories.

The first strategy is a depth-1 tree search (i.e. reflex policy). It has been implemented as a baseline to explore this strategy. Increasing the depth however is unfeasible given the size of our action space.

The second strategy we use is fitted value iteration with stochastic sampling from our model (which is entirely deterministic). The value function was thus written as $V(s) = \theta^\intercal \phi(s)$ where θ will be the vector of parameter we will try to fit and ϕ is a mapping function we will be discussing in Sec. 3.2. Two strategies have been attempted to update the value of θ as the sampling is done. The first one is gradient descent with regularization, and the other is batch least squares every given number of samples.

2.3 Reward function

To fully achieve balance of the pendulum on the quadcopter we must account for several factors in our reward function. First, the drone flight must be stable. This means that the position of the drone

in space should remain steady. Additionally, we must account for the pendulum being balanced, i.e. the displacement of its center of mass relative to its base should be close to zero. So far we have experimented several different expressions for a reward function with these characteristics.

The reward has been initially defined as $R = -\|r - r_{ref}\|^2 - \|E\|^2 - \|\dot{r}\|^2 - K\left(\|\rho\|^2 + \|\dot{\rho}\|^2\right)$ with $r = [x, y, z]$ the position vector of the drone, $E = [\alpha, \beta, \gamma]$ the attitude of the drone (using Euler angles), $\rho = [a, b]$ the position of the center of mass of the pole with respect to the center of mass of the drone and K weighting factor.

3 Preliminary experiments and current progress

3.1 Reflex policy: depth-1 tree search

A reflex policy has been implemented to serve as a baseline. Given the current state, we loop over all the possible actions and pick the action that will maximize our reward (defined in section 2.3).

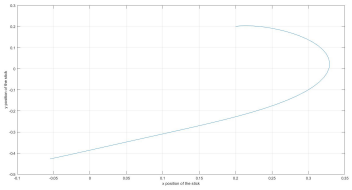

Figure 1: Trajectory of the stick's center of mass with reflex policy

As we can see on Fig. 1, the reflex policy does not stabilize the drone and the stick, although it does seem to move in the right direction initially to try to achieve this.

We also tried depth-2 tree search, but because of the elevated number of actions and the exponential growth of the search space, computation was too slow to practically implement.

3.2 Early Value iteration attempts

The first value iteration algorithm we built used fairly straightforward mapping function. The idea was to use second order polynomials. Thus it includes an intercept term, the step itself, and possible second order combinations of two features.That is, we defined our feature mapping as follows:

$$\phi(s)_{14(i-1)+j} = s'_i \cdot s'_j \quad \text{for} \quad s \in S, s' \in \mathbb{R}^{14}, s' = (1, s)^\intercal, i, j \in \{1, 2, ..., 14\}$$

Early attempts at solving the problem were unconvincing. It performed similarly to reflex policy most of the time, that is the initial movement was usually in the right direction.

The first major iteration performed on the system was attempting to improve the mapping function. The major oversight in our first approach was using the angles directly, as there is no reason to believe the value function would map quadratically to angle values. However, it should be related to the direction of the thrust vector that those angle determine. Therefore we have replaced the angles in the mapping function with the coordinate of the normalized acceleration vector. Additionally, since our problem is symmetric, we have experimented with removing the linear term.

As of this milestone, the only successful attempt to balance the pendulum was the result of reward hacking. The program had discovered in a few of our simulation that shooting downwards was a way to stabilize the pendulum, and thus it simply propelled itself downwards, which balanced the pendulum for a while. Though this solution is clearly unfeasible in practice, it was the optimal policy when altitude error was barely penalized. The trajectory of the center of mass during one of these attempts is shown below:

3.3 Error Analysis

So far we have mostly explored different strategies to compute the best action for each possible state. We are assuming that our simulator for the dynamics of the system is accurate and models reality precisely, since its implementation matches the one described in the article by Hehn [1].

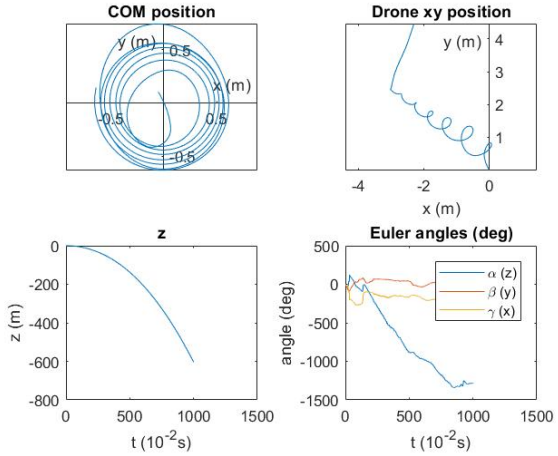

Figure 2: Trajectory of the center of mass of the stick (top left). The other three plots show the position of the drone in 3-D space over time

From what we saw in lecture, we would need a human baseline θ_{human} against which we could compare the value of the reward obtained by our policy. If the reward obtained by latter was greater than the reward obtained by the human baseline, we would know that the problem lies in the way we are defining our reward (since maximizing the reward is not yielding expected behaviour). If, on the other hand, the human baseline resulted in much higher reward, the problem would be with our implementation of our RL algorithm.

The problem of error analysis for us lies in obtaining a human policy for our whole system, since even for a skilled drone pilot, balancing an inverted pendulum on a drone would be challenging. One approach we decided to try is to first decouple the pendulum from the drone, and test our RL algorithm on a reward function that represents stable flight and compare that to a human baseline. As our control input is the thrust of the propeller and the rotation of the drone to thrust in the right direction, the problem is far from trivial. For this problem we have simply reduced our reward function to $||r||^2 + ||\dot{r}||^2$ and the feature mapping function was a polynomial in r,$\dot{r}$ and u (the thrust direction). Our first attempt was successful in keeping the drone within a few meters of the origin without a drop in altitude for about a full minute. This is quite an achievement based on the current control scheme. A next natural step is to couple the pendulum and focus on adding terms that represent stabilization of the inverted pendulum to the reward function.

4 Future work

4.1 Comments on difficulties faced so far

After this first phase of experimentation, one main difficulty has been made clear: one of the biggest obstacles to successful flight is the unconventional control scheme. Our actions are not directly related to position or velocity but to the orientation of the state vector, making the link between the 2 more obscure. One of our future objective should therefore be to find a way to control the drone itself reliably.

4.2 Where to go next

It is clear two of the main problems we are facing are inadequate reward functions and feature mappings. Having a quadratic (i.e. based on the norm) reward function means that the value function was more complex, making less realistic to model it using a quadratic function. Experimenting with step rewards function might be more efficient in the future. Furthermore, adapting our feature mapping to the problem might be a huge gain in efficiency. For example, we could exploit the symmetries of the problem, both with regards to the origin (allowing us to drop the linear terms of the mapping) as between the different variables (the coefficients should be the same for terms in x and y).

References

[1] Markus Hehn and Raffaello D'Andrea. A flying inverted pendulum. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 763–770. IEEE, 2011.

[2] Samir Bouabdallah and Roland Siegwart. Full control of a quadrotor. In *Intelligent robots and systems, 2007. IROS 2007. IEEE/RSJ international conference on*, pages 153–158. Ieee, 2007.

[3] Kangbeom Cheon, Jaehoon Kim, Moussa Hamadache, and Dongik Lee. On replacing pid controller with deep learning controller for dc motor system. *Journal of Automation and Control Engineering Vol*, 3(6), 2015.

[4] Angela P Schoellig, Clemens Wiltsche, and Raffaello D'Andrea. Feed-forward parameter identification for precise periodic quadrocopter motions. In *American Control Conference (ACC), 2012*, pages 4313–4318. IEEE, 2012.

[5] Markus Hehn and Raffaello D'Andrea. Quadrocopter trajectory generation and control. *IFAC Proceedings Volumes*, 44(1):1485–1491, 2011.

[6] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1642–1648. IEEE, 2010.

[7] Rafael Figueroa, Aleksandra Faust, Patricio Cruz, Lydia Tapia, and Rafael Fierro. Reinforcement learning for balancing a flying inverted pendulum. In *Intelligent Control and Automation (WCICA), 2014 11th World Congress on*, pages 1787–1793. IEEE, 2014.

[8] M Sedighizadeh and A Rezazadeh. Adaptive pid controller based on reinforcement learning for wind turbine control. In *Proceedings of world academy of science, engineering and technology*, volume 27, pages 257–262, 2008.

[9] Xue-Song Wang, Yu-Hu Cheng, and Sun Wei. A proposal of adaptive pid controller based on reinforcement learning. *Journal of China University of Mining and Technology*, 17(1):40–44, 2007.

[10] Shin Wakitani and Toru Yamamoto. Design and application of a data-driven pid controller. In *Control Applications (CCA), 2014 IEEE Conference on*, pages 1443–1448. IEEE, 2014.