Automated Aircraft Touchdown Using Reinforcement Learning

Rahul Sarkar, Amy Shoemaker, Sagar Vare

1 Introduction

Automated flight control systems are installed in all military and commercial aircraft today due to their superior collision avoidance capabilities, their ability to auto-pilot the aircraft under mild weather conditions, and their ability to act as a check against piloting errors. In recent years, reinforcement learning (RL) has been used to design controllers for unmanned aircraft vehicles [1].

While these systems have tremendously increased the safety of air travel by reducing accidents, particular attention is now being paid to take-off and landing, as these are two of the most critical phases during a flight's journey from the point of view of air safety. Hence, there exist automated landing systems (ALS) designed to land aircrafts with high reliability. RL techniques have also been applied to this particular task of landing aerial vehicles. Vision based algorithms have been shown to be capable of helicopter landing and of performing automatic detection of landing sites [4]. RL-based algorithms have even been used to demonstrate how in the future one could perform efficient and accurate spacecraft landings on a planet like Mars, without the need for an offline trajectory [2].

A major issue, however, is that the ALS disengages when the flight conditions are beyond the preset parameter range of the ALS system. This frequently happens in turbulent weather conditions, thereby forcing the pilots to perform a manual landing, which presents a great risk especially for pilots with limited experience and due to limited visibility. Thus there is a need to increase the operational range of the ALS systems. This problem has been examined using recurrent neural networks and genetic algorithms, guiding automated controllers to successfully overcome wind disturbance during the landing stage of an aircraft [3]. We are interested in further investigating the potential for safe, RL-based ALS in the face of high and variable winds.

2 Problem Definition and Measures of Success

We plan to implement an RL agent that guides an aircraft in its final descent phase (final 5km up to touchdown), during potentially windy weather. We consider a simplified model, where there are only two dimensions that the RL agent needs to control, namely the lateral velocity v_y and the vertical velocity v_z of the aircraft (motion along the x-direction is predefined). The unpredictability of the crosswind, especially during stormy weather, leads to sudden lateral changes in the aircraft's position and velocity that causes passenger discomfort. At each time step, the aircraft has a set of possible actions that it can take, which changes its lateral and vertical velocities. While executing the action, the RL agent must also take into account the effect of the wind's deflection from the aircraft's planned trajectory.

Success of the descent is determined by safe landing and by the level of passenger comfort. In considering safety, we require that our aircraft lands near the middle of the runway at a very minimal horizontal and vertical velocity. To minimize passenger discomfort, we look to minimize jerk and extreme acceleration throughout the entire journey.

3 Approach

Even in considering the two-dimensional version of this problem, the state space is still extremely large, because the RL agent has to control both position and velocity along these two directions. We thus look to discretize the underlying state space of positions y and z, velocities v_y and v_z, and wind speed v_w, which are our state variables (note that this leads to a breakdown of the Markov assumption). We create a predefined number of discrete bins for each state variable, and assign the actual state to the center of the bin it belongs to. In this way, discretization leads to a lack of precision, which can be interpreted as noise associated with sensor measurements of the state and action space. We are currently considering $|v_y|_{\max} = 50$ km/hr, $|v_z|_{\max} = 20$ km/hr, $|y|_{\max} = 0.25$ km, $z_{\max} = 1$ km, $z_{\min} = 0$ km, and $|v_w|_{\max} = 25$ km/hr. If the aircraft goes beyond these limits, we consider it as an end state and give a large penalty.

At each step we have a penalty based upon the jerk and the acceleration of the aircraft in any particular direction, which model the level of discomfort the passengers feel. We will include in our model a penalty for actions involving extreme acceleration in the y or z directions. To ensure safety, as defined in Section 2, we also have a penalty if the position of the aircraft strays far from the center of the runway in the lateral direction.

The aircraft receives a final penalty based upon its landing position, which is zero if the aircraft lands in the center on the runway, and steeply increases away from the center.

Wind acts like an adversarial phenomenon for our RL agent, as there is a stochastic component to it. To account for this, we model wind as a Markov process with a normal Gaussian distribution for its transition probabilities from one time step to the next. If the wind velocity is v_{w_t} km/hr at time t, then $v_{w_{t+1}} = \mathcal{N}(v_{w_t}, \sigma)$ for some given parameter, σ. Setting a large initial v_w and/or a large σ can help simulate stormy weather.

In considering an algorithm for initial implementation, we decided on using Q-learning because of its online learning nature, giving it a chance to interact with the simulator that we have coded up. Even with discretization however, we are faced with a very large state space ($100 \times 100 \times 100 \times 275 \times 40 \times 50$) and action space ($100 \times 100$). This makes any standard learning approach unfeasible without some form of function approximation. Our first attempt will thus be to implement a Q-learning algorithm with function approximation. Considering the size of our action space, we also plan to experiment with Sarsa(λ), which unlike Q-learning, does not have to choose a maximum over the whole action space in each update of $Q(s, a)$. We will of course include function approximation in our Sarsa(λ) implementation, as well.

Additionally, we plan on working with policy search algorithms, such as cross entropy method, based upon some parameterization of the policies. While it is known that the state and action space is large, the policy space might be in a lower dimensional space, and it would be interesting to explore this.

4 Current Progress and Next Steps

We have already completed a basic simulator that can interact with our RL agent. Given a set of initial parameters (initial state, σ for the wind's Gaussian distribution, discretization parameters, etc.) and an action declared by the agent, the simulator determines and logs the subsequent state before discretizing. We use a first-order integration scheme to update the position and velocity of the aircraft along both directions inside our simulator, that also incorporates the effect of wind. When the RL agent asks for the next state, the simulator returns the discretized state value.

While our simulator currently captures the unpredictability of the wind using a Gaussian distribution to update the state of the wind at the next time step, it does not take into account the inaccuracy of sensor data. While technological advances in positioning systems have led to high accuracy in position sensor data, discrepancies between weather sensor data and actual weather effects can be significant [3]. As an extension of our simulator, we would like to model this discrepancy by incorporating stochasticity into the effect that current wind reading v_{w_t} has on the subsequent horizontal position y_{t+1} and the subsequent horizontal velocity $v_{y_{t+1}}$.

Another area of potential extension is to consider the plane's forward motion v_{x_t} and position x. For simplicity, we have restricted our model to two dimensions (altitude, z, and lateral position, y), implying a pre-determined forward velocity v_x for each t. In the future, we would like to implement a three-dimensional model, in which our agent learns a policy that includes Δv_x as part of each action. In the same vein, it would be interesting to extend our simulator by considering a more robust model of the wind. In our current simulator, wind is only experienced laterally along the y direction. However, in further research we would like to lift this restriction.

References

[1] Haitham Bou-Ammar, Holger Voos, and Wolfgang Ertel. Controller design for quadrotor uavs using reinforcement learning. In *2010 IEEE International Conference on Control Applications*, pages 2130–2135. IEEE, 2010.

[2] B. Gaudet and R. Furfaro. Adaptive pinpoint and fuel efficient mars landing using reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 1(4):397–411, Oct 2014.

[3] Jih-Gau Juang, Hou-Kai Chiou, and Li-Hsiang Chien. Analysis and comparison of aircraft landing control using recurrent neural networks and genetic algorithms approaches. *Neurocomputing*, 71(1618):3224 – 3238, 2008.

[4] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme. Visually guided landing of an unmanned aerial vehicle. *IEEE Transactions on Robotics and Automation*, 19(3):371–380, June 2003.