CS 5013: Spring 2023
Air Traffic Control Simulation

Sam Bird

Abstract

openScope is an air traffic control simulator that simulates the work of an approach/departure controller, variously referred to as Terminal Radar Approach Control (TRACON), Approach, or Director, as well as a tower controller. openScope provides a sandbox in which we can simulate the behaviour of several AI reinforcement learning agents attempting to safely navigate arriving aircraft towards an airport.

Keywords: air traffic control, reinforcement learning, artificial intelligence

1 Introduction

Air traffic controllers are responsible for moving aircraft safely towards their destination. They need to ensure aircraft do not collide with each other, as well as ensure that aircraft reach their intended destinations. To that end, they are assisted by *airways*, which provide predictable paths for aircraft to follow, as well as technology to warn the controller when two aircraft are too close to each other, or are on a path that would lead to a loss of separation [1].

These challenges are known in aviation jargon as *separation* and *sequencing*. Separation is the physical separation of one aircraft from all other aircraft; consequently a "loss of separation" is when an aircraft becomes dangerously close to, or collides with, another. Sequencing is quite like assembling a queue: it refers to guiding aircraft into landing, one after another, onto the runway.

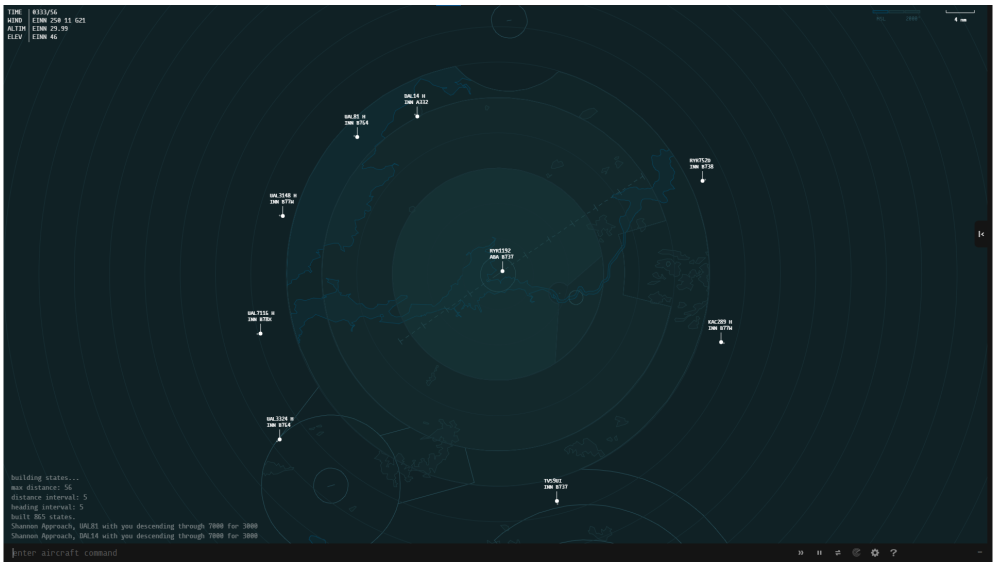
There are several reasons why AI can't perform the tasks expected of a human approach controller. Human controllers must be ready to deal with challenges that arise in-flight, for example if an emergency occurs on board an aircraft. They must be able to communicate with pilots and understand any situation that may arise. Perhaps most importantly, aviation is one of the most regulated industries in the world. AI is not a serious option being considered by aviation authorities in the medium-term future [2].

However, openScope gives us a sandbox to use reinforcement learning techniques to see how aircraft would vector themselves if left alone. By providing a model of aircraft and airspace around an airport, I wrote reinforcement learning agents to sequence themselves into landing.

2 Methodology

I use openScope [3], an open-source air traffic control simulator, to run multi-agent reinforcement learning scenario to move arriving aircraft towards a position where they could be cleared for approach. openScope did not provide any AI or reinforcement learning libraries: those capabilities were entirely implemented by me.

2.1 Overview



```
TIME  | 0333/56
WIND  | EINN 250 11 G21
ALTIM | EINN 29.99
ELEV  | EINN 46
```

```
building states...
max distance: 56
distance interval: 5
heading interval: 5
built 865 states.
Shannon Approach, UAL81 with you descending through 7000 for 3000
Shannon Approach, DAL14 with you descending through 7000 for 3000
```

Welcome to Shannon Airport (EINN), the third busiest airport in the Republic of Ireland. Shannon is a good case study because of its relatively circular airspace and the fact that aircraft arrive from all directions. Winds are currently blowing out of the southwest at 11 knots, meaning that arriving aircraft will be landing on runway 24 today.

Each agent will be performing Q-learning. Q-learning assigns a Q value to each possible action that can be taken in each state. Sometimes, the agent will (with probability ϵ) explore by taking a random action. Otherwise, it will take the best action available to it: that whihc maximises utility.

Q-learning agents learn by taking a sample and observing what happens when taking action a at state s. It learns at a rate α, augmenting the current Q value by that factor.

$$\text{sample} = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha \cdot \text{sample}$$

2.2 States

The airspace (and the area beyond the airspace) is divided into states, segments centred at the origin and extending around in 5-degree arcs.

2.2.1 Terminal states

Aircraft may reach one of two terminal states. The first is being cleared for approach: if the aircraft may be cleared for approach, it receives a reward of 1000.

Aircraft do not intentionally move directly downwards, so the reward isn't given when the aircraft is directly over the runway. Instead, it is given when the aircraft reaches a position in which it could make a successful approach, descending onto the runway. There are three criteria that an aircraft must satisfy to request approach clearance:

1. The aircraft is more than 15 miles from the runway (so it has time to descend) but not more than 30 miles from the runway (so that it isn't too early to request clearance)
2. The aircraft is within 20 degrees of the runway heading (so that we know it isn't on the other side of the airport)
3. The aircraft's heading is within 20 degrees of the runway heading (so that we know it is flying towards the runway)

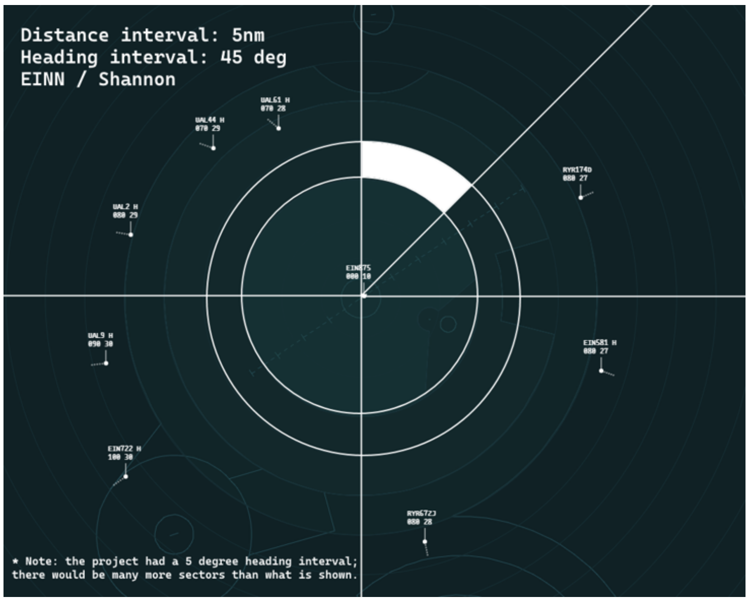
Satisfying these three criteria will allow the aircraft to request approach clearance. At this point, the agent is terminated and no more actions are taken.

The second terminal state is when the aircraft leaves approach airspace. Keeping aircraft within our (Shannon Approach's) airspace is very important: this is the area we are authorised to move aircraft around in. If an aircraft leaves our airspace, it is in an area that belongs to Shannon Control, meaning that there could be other aircraft flying there that we are unaware of. For safety reasons, it is very important we keep aircraft inside our airspace. Once an aircraft leaves our airspace, we can no longer control it (the agent terminates) and a penalty of -1000 is given.

2.2.2 Nonterminal states

The scope is divided up into different states. Each state is defined by the distance from the centre of the scope and the degrees it starts at; in this way, it may be thought of as a polar co-ordinate system. For example, the state highlighted in the image would be (10, 0) because the lower bound is 10 nautical miles away from the airport and it starts at 0 degrees, due north. In the image, the state extends for 45 degrees, but in the actual simulation each state is just five degrees.

We need to divide the scope into various states because openScope does not do it for us. Internally, every position stored is a pair of co-ordinates: latitude and longitude. This means the state space would be continuous, not discrete: each "state" could only be differentiated by the floating-point error of the JavaScript engine on which it runs,



so observing transitions for every state (and storing them in memory) would be an impossible task.

There is a drawback in coming up with our own states like this: the farther away we get from the airport, the bigger each state is. This doesn't end up mattering very much, because most outer states can be summarised as ”moving towards the airport is good, leaving the airspace is bad” - there does not need to be as much precision in the outer states as there is in the inner states.

2.3 Actions

The actions in this simulation are very simple. In each nonterminal state, an agent may move either north, east, south, or west. These actions are not instant and are the equivalent of instructing a pilot to fly a heading of 360, 90, 180, or 270 degrees, respectively. The aircraft will turn to fly the instructed heading and will maintain that heading until it enters the next state and it receives a new instruction.

If an aircraft enters a terminal state where it is possible to conduct an approach, its action will be to request approach clearance. An aircraft entering a terminal state by exiting the airspace will conduct no further action; in fact, it ceases to be controllable.

Actions are not probabilisitic: they are successful and obeyed 100% of the time.

3 Result

3.1 Experimental setup

3.1.1 States

As mentioned before, states divide the airspace around the airport. In this simulation, a new state begins every 5 degrees around the airport, and every 5 nautical miles from the airport.

Each state stores the Q-values of each action that is possible from that state. Each agent maintains a copy of which state it is currently in.

3.1.2 Agents

Agents are very simple: they maintain a reference to the aircraft that they are flying, the state they are currently in, and each agent is able to check if they are currently capable of conducting an instrument approach.

Agents may take any legal action. It is important to note the difference between a legal action and a penalised action. In particular, leaving the airspace is a legal action, it merely incurs a sharp penalty. This is to say that the state space is bigger than the controllable airspace, because leaving the controllable airspace is legal, but will put the aircraft in a terminal state.

3.1.3 Observing state transitions

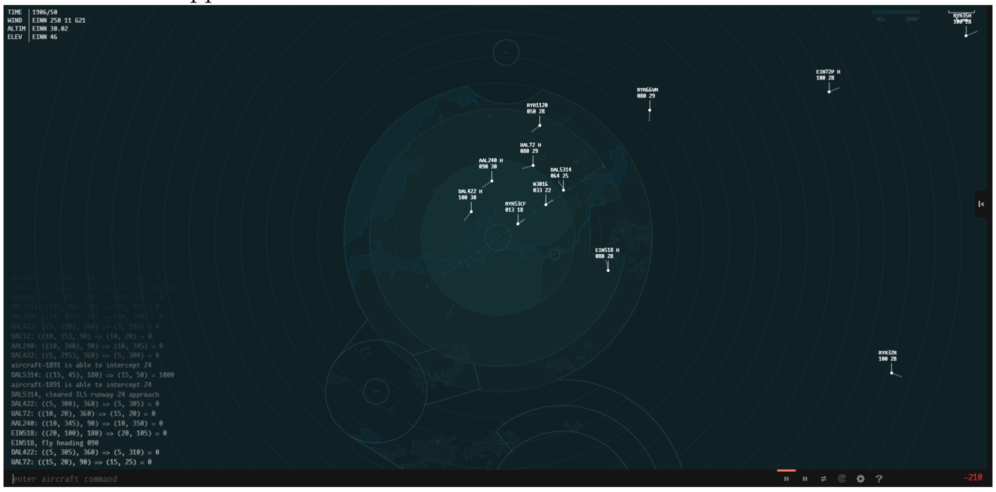
Aircraft update their positions every tick. Every tick, we check to see if an aircraft has passed from one state to another. If the aircraft changes state, we will observe its transition.

```
let reward = 0;

if (agent.canConductInstrumentApproach(agent.aircraftModel.positionModel,
                                        this.arrivalRunway)) {
    reward = 1000;
} else if (!agent.aircraftModel.isInsideAirspace(this.airport)) {
    reward = -1000;
}

this.transition(agent.lastState, action, agent.nextState, reward);
```

The transition function is simple: it updates the Q-value associated with the present state and action with the reward learned by taking this action at this state.

3.2 Results

Agents generally learn to sequence themselves and to fly towards a position that allows them to conduct an instrument approach. They also learn to avoid flying outside of controllable airspace. If the learning rate is turned down to 0 and all agents fly based solely on the learnt Q-values (after about 12 hours of learning), we see that all agents will form a relatively orderly queue and proceed to a point where they can be cleared for an instrument approach.



```
TIME  | 1906/50
WIND  | EINN 250 11 G21
ALTIM | EINN 30.02
ELEV  | EINN 46
```

```
DAL422: ((5, 285), 360) => (5, 290) = 0
N3016 intercepted localizer above glideslope
AAL240: ((5, 335), 360) => (10, 335) = 0
DAL5314: ((15, 40), 90) => (15, 45) = 0
AAL240: ((10, 335), 90) => (10, 340) = 0
DAL422: ((5, 290), 360) => (5, 295) = 0
UAL72: ((10, 15), 90) => (10, 20) = 0
AAL240: ((10, 340), 90) => (10, 345) = 0
DAL422: ((5, 295), 360) => (5, 300) = 0
aircraft-1891 is able to intercept 24
DAL5314: ((15, 45), 180) => (15, 50) = 1000
aircraft-1891 is able to intercept 24
DAL5314, cleared ILS runway 24 approach
DAL422: ((5, 300), 360) => (5, 305) = 0
UAL72: ((10, 20), 360) => (15, 20) = 0
AAL240: ((10, 345), 90) => (10, 350) = 0
EIN518: ((20, 100), 180) => (20, 105) = 0
EIN518, fly heading 090
DAL422: ((5, 305), 360) => (5, 310) = 0
UAL72: ((15, 20), 90) => (15, 25) = 0
enter aircraft command
```

Most interestingly, agents seem to manage risk. While a human pilot might fly well into our airspace to avoid inadvertently leaving it, the Q-learning agents seem content to fly relatively close to the airspace boundary. They have learnt that there is no penalty for flying close to the edge, merely for flying outside of it. With moves deterministic so that there is no risk of an action leading to a different state than the one intended, agents aren't penalised for flying close to the border. In this sense, there could be benefits found in the amount of space available to controllers.

Ultimately, however, the fact that these aircraft managed to sequence themselves does not resolve the greatest challenges in controlling air traffic. It does not account for separation, nor could agents in this experimental setup learn to avoid one another, not just staying inside our airspace.

4 Conclusion

I created Q-learning agents to learn to sequence themselves for approach into Ireland's Shannon Airport. By rewarding aircraft that entered a state where an approach clearance could be received and penalising aircraft that left approach control's airspace, agents found a way to line themselves up to land while staying inside the airspace.

However, more work is needed to address the challenges that come with air traffic control before we find AI in control towers any time soon. In future, AI will need to be able to recognise when agents are about to come into conflict with each other and direct them away. More actions will need to be added to give greater flexibility to controllers instead of the four cardinal directions that this experiment limits itself to. Finally, we will need to account for altitude in future, as this simulation simply assumes that aircraft will descend to a suitable altitude by the time it requests an approach clearance, which is certainly not true of all airports.

5 Task assignment

This was a solo project; I completed all tasks.

References

[1] Federal Aviation Administration: Terminal Automation Modernization and Replacement (TAMR). Accessed 27 March 2023. (2022). https://www.faa.gov/air_traffic/technology/tamr

[2] Smith, L.: The challenges of fully automating air traffic management. NATS. Accessed 27 March 2023. (2020). https://nats.aero/blog/2020/08/the-challenges-of-fully-automating-air-traffic-management/

[3] openScope. Accessed 1 April 2023. https://github.com/openscope/openscope