CS 181 Final Project:
AI-ATC: Artificial Idiot Air Traffic Controller

Hu Hang, Ye Jingjing ,Diao Zihao, Hou Jingyang

[1] School of Information Science and Technology,
ShanghaiTech University, Shanghai 201210
[2] {huhang, yejj,diaozh, houjy }@shanghaitech.edu.cn

Abstract. Air traffic control (ATC) is a service provided by ground-based air traffic controllers who direct aircraft on the ground and through controlled airspace, and can provide advisory services to aircraft in non-controlled airspace. The primary purpose of ATC worldwide is to prevent collisions, organize and expedite the flow of air traffic, and provide information and other support for pilots. Human air traffic controller provides service worldwide. But when human get distracted by other factors, accidents follows. In fact, many research suggested that the major factor that causes accident is the human factor (4). What we do in this project is trying to use the artificial intelligence approach to eliminate the problems caused by distracted controllers. In this project, we built two simplified models: one for the airplanes and another for the approach area with a set of rules compatible with *Civil aviation air traffic management regulations*(CCAR-93TM-R5:) of the Civil Aviation Administration of China (CAAC) to train a model that could direct the air traffic.

1 Introduction

The air traffic control is a complicated system that works in almost all parts of civil aviation including the regulator (e.g. Federal Aviation Administration), carrier (e.g. China Eastern) and airports (e.g. San Francisco International Airport). The factors a controller should concern would include the status of airplane, airport capacity, the weather condition and many other things. Controller also needs to cooperate with their peers in different airports, air traffic control areas and even countries. However, this project will work on a simplified model consists of a single airport, a single stage of air traffic and with no special weather condition and emergency situation. But we believe that in a more realistic and complicated situation, the principle of this problem will not change.

This project will focus on modeling the traffic of Chengdu Shuangliu International Airport that serves Chengdu, the capital city of southwest province of Sichuan. As the major airport of a city with little railway and highway connection compared with its economic development, the airport with two runways ranked 4^{th} by the traffic in all airports in mainland China in 2017(2).

Also, this project would only deal with only one stage of ATC, approaching.

Our simulation program would have three parts what works asynchronously and communicate with each others using Python's built-in multiprocessing.Queue data structure. The first part of this program is the environmental models where the terrain and airport location is specified. The second part is the airplane models. This subsystem has two parts. One for the airplanes' specification and another one would fetch the real time air traffic data from the internet. The most important part is the part where the directing task is carried out.

2 Design of Environmental Models

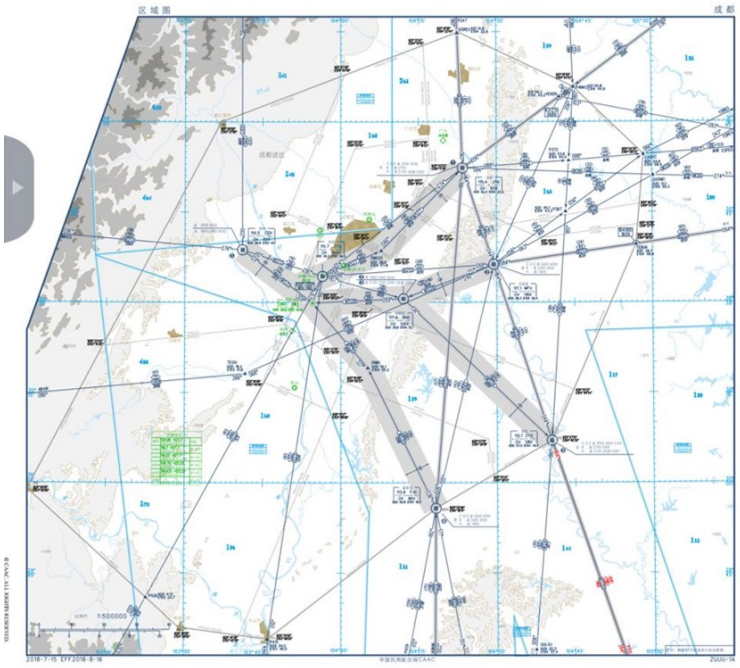

Fig. 1. Polygon region of Chengdu Shuangliu Intl. Airport. Source: CAAC Southwest

The aim of this subsystem is to provide a comprehensive environmental model consists of the terrain data, airplane runways and interval control of airplanes.

The real world ATC works in a "continuous" way. The airplanes could be at everywhere they are allowed to be and the the airport's runway would vary in length and direction. But in practice, controllers would divide the airspace into multiple flight levels vertically, and the airplanes are not allowed to get too close to each other. Those constraints allow us to simplify the environmental model into a discrete one.

We divide our approaching area into grid of size 2500 meters $\times$ 2500 meters $\times$ 300 meters. The horizontal size of the grid is 2500 meters and the vertical size is 300 meters. The numbers come from the CCAR-93TM-R5 of CAAC, 300 meter is exactly the vertical size of a flight level and 2500 metes is the minimum horizontal distance of two airplanes(1). We set a constraints that only one airplane is allowed in each grid.

For the terrain data, we downloaded the 200-meter topographic map of Chengdu and map it two our grid by taking the average of altitude of the middle point and four vertex of the grid. That is:

$$\text{altitude}_{\text{grid}} = \frac{1}{5} \sum_{\text{middle point and vertex}} \text{altitude}$$

If a gird is lower than the altitude of the terrain, airplanes are not allowed in this area.

The runway is simplified as a single grid in the environmental model. If an airplane is entering this grid with the right heading and altitude, our program would consider it as landed. When the flight is 20 kilometer far way from runway and about 1 kilometer in altitude, The controller will hand over to tower. More detail about hand out is in the "Agents and Actions" part.

Another simplification is done specifically for Chengdu Shuangliu Airport. The approach area of the Airport is a polygon one as shown above. We simplify the polygon area to a rectangle one. The longitude and latitude of this vertex of this region is (102.28°E, 31.13°E), (102.28°E, 29.51°N), (106.22°E, 29.51°N), (106.22°E, 31.13°N).

3 Design of Airplane and Traffic Models

The airplane model is where the sates and specification of the airplanes are stored. There are two parts in this subsystem. The first one is the airplane specification where provides the ATC with parameters like the speed limit and altitude limit of the airplane so that the ATC (agent in this project) could provide proper instruction to the pilots. The second one is the real time data crawler that fetch air traffic from the internet data sources.

The first part of airplane models would provide the following information to the agent that carry on the air traffic control. The data this model provides include:

- Airplane Type, in the ICAO aircraft type designator format, e.g. A320;
- Manufacturer, e.g. Airbus;
- Maximum speed of this type;
- Maximum altitude of this type;
- Maximum weight of this type.

Those information is fetched from the manufacturer's website in advance and stored in our local file system.

The second part is the real time traffic data crawler. This part would get data from a website provides real time air traffic data called FlightRadar24 (https://www.flightradar24.com/). This website provides data of the automatic dependent surveillance broadcast (ADS-B) broadcast of airplane. ADS-B is a technique for the ATC to track where the airplane is. It relies on the airplanes to broadcast their location, airplane type, airline, flight number and other information at a certain frequency in a certain time interval (FAA). All signals are clear and can be received and decoded using a decent receiver. This website has a batch of volunteers worldwide to use their own receiver to receive those signals, decode them and transmit it to the server of this website and then to other users around world using web technology.

Those data would serve as the training data for our agents. Our program would fetch traffic data from this website and try to construct our own airplanes within the approaching control area. More specifically, we get data from FlightRadar24 in a certain time interval, construct model for those airplanes just entering our ATC area combined with the data we get from the airplane models, then add the models to the environment for the agents.

4 States and Actions

4.1 States

To describe an airplane and compute its movements in the model built above, the following parameters are needed:

- Coordinate (X and Y, latitude and longitude, in degree)
- Altitude (elevation, in meter)
- Ground speed (in m/s)
- Heading (in degree, with North)

4.2 Actions

For coordinate, it cannot be directly changed in our project (also in the real world), thus we can only compute them from the ground speed.

The original coordinate is from the data source. From then on, we compute the coordinate given speed and heading. Though the earth is a sphere, we simplified it by calculating in straight lines on a flat surface instead of arcs on a sphere.

The altitude is not easy to compute from other states, but we notice that in the actual controlling process, the minimum height change in an single instruction is 300m. Therefore, the actions for altitude are as follows:

- Up (go up 300m in altitude)
- Down (go down 300m in altitude)
- Stay (do not go up or down)

The speed of a plane is a continuous data, in order to make the number of actions finite, we set the difference in speed between two continuous states as fixed numbers 8, 0 and -8:

- Speed down for 8 m/s
- Speed up for 8 m/s
- Do not change speed.

The heading is also originally continuous, but that would lead to infinite actions. Thus considering the airplane's ability to turn, we choose 5 discrete actions for turning as follows:

- Turn left 30 degree
- Turn left 60 degree
- Turn right 30 degree
- Turn right 60 degree
- Go straight

An action in this project is a combination of the previous actions, these factors make it possible for us to compute the next states, which help in building the learning model.

5 Q-learning algorithm

5.1 Infeasibility of Model-based learning

Although the transition function from one state to the next can be calculated based on the airplane location and the the action it takes, but computing the transition value at each time step for each possible state involves a huge computational complexity.Moreover, since the state of a model-based learning represents the situation of the whole space at a given time step, when the number of airplanes inside the control region increases, each additional plane will lead to a exponential increases on state space and action set. Therefore, a model-based learning can not serves as a fine algorithm in air traffic control system.

5.2 Applying Q-learning

Before applying Q-learning, we need to specify our state space and the action set, it is without doubt that the air control region is a continuous space, and there a infinite many actions the plane can take. So the first thing we need to do is to discretize our space and action. Based on CAAC official documents, We finally get a $71 \times 105 \times 17$ states' state space and a $5 * 3 * 3$ actions' action set. (The detail of implementing the discretization have been stated in the *States and Actions* part.)

Reward We set our reward function manually instead of learning it from the real world data, this is because the real world trajectory follows a settled pattern, and the pattern is fixed for easier controlling of the human air traffic controller. But since we are AI ATC, we can find a better trajectory without the fixed pattern.

- We do not want any of the two planes collide with each other, safety always has the highest priority, so we give collision the highest penalty to make sure no plane will choose an action taking the risk of collide with another plane. Based on the CAAC official document, we find that a distance within 3.5kilometer will be regarded as having the potential of collision. We set the distance of the diagonal of a state to 3.5 kilometer, we can then judge whether two planes collide with each other by observing whether they are within the same state.
- The plane is supposed to take off gradually with a decreasing speed in normal cases. So we give both lower altitude and lower speed a higher reward.
- Since the aim of each plane is to landed on the runway, so it is important that the plane flight closer to the runway when it flights at a lower altitude, so the reward for flying close to the runway at a low altitude will be punished if it is too far from the runway, but also we do not want all planes to circle over the runway, so the plane at a high altitude will be punished for flying to close to the runway.
- The flight should be heading to the runway when it is passed to the hand over point, so only the planes with the correct heading can take an action to go to the hand over point, other planes can not reach to the hand over point even if the runway is empty, and it is close to the runway.
- We also want the planes to land at a shortest time, so we give the plane a living penalty to make sure it do not want to life long.
- We set the runway to be occupied for two minutes when a plane reach the hand over point, within two minutes, other planes can not get to the runway even if all other conditions are satisfied.

Learning Process We start at the first detected point of the plane in the control region, and generate it's best action by first learning through 1000 steps ahead to see which action can gives the highest reward. The next state of an action wad defined by the continuous position of the plane and the action it will take. The Q-value of the action is updated with the following function:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,\pi(s),s') + \gamma V^{\pi}(s'))$$

After going through the training process, we actually determined the best action we can take by choosing the highest reward action we can get at the current state by updating $Q(s,a)$ again.

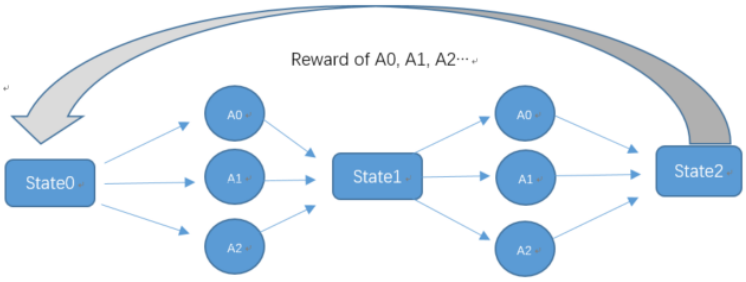

Fig. 2. predict two steps to get the reward

Each plane is an individual Q-learning agent, they learn the best action by regarding other planes as environmental data. It is possible that two agents choose the same state as the next state, when this happen we will ask the agent who gets a lower reward to take a sub-optimal action.

6 Result

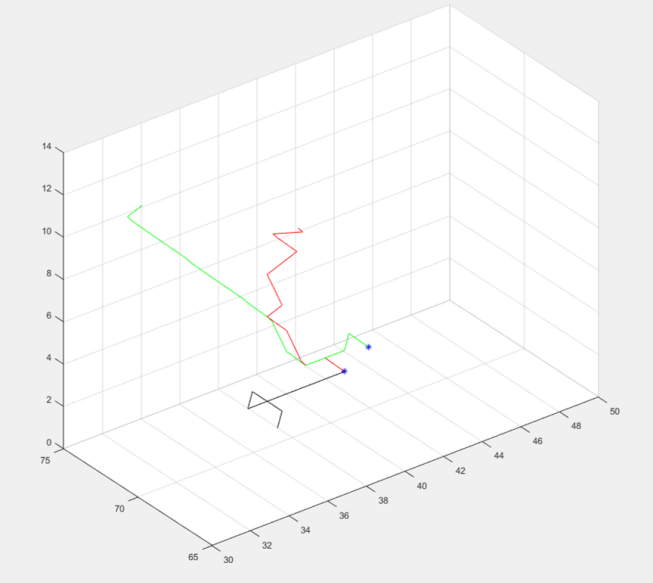

Fig. 3. the trace from the state points we get of 3 flights

We managed to find trajectory for all airplanes within the control region without any collision, and all of them can head for the hand over point with a correct heading and an appropriate speed, the result shown above gives the trajectory we found for all planes inside the region for a short period. Because we eliminated time axis in the above picture, so the trajectory seems overlap with each other, but in fact, after adding time axis, no collision will happen.

7 Conclusion

7.1 Completed part

The ATC task is completed in a simplified environment by our agent. We run the code for almost 24 hours. Every flights have landed successfully without collision and hasn't make blockage.

7.2 Disadvantage

However, our simulation is done under a simplified condition. In real world, the problem is more complicated. For example, the agent need to consider the priority of the airplane (so that the airplane with emergency situation on-broad could land first).

Also our algorithm sometimes has a oscillating output. It tend to change the direction and speed of an airplane in almost every time period. This behavior is not feasible because it will require pilot's action every minute and cause extra turbulence to passengers.

Bibliography

[1] CAAC (2017). *CCAR-93TM-R5: Civil-aviation air traffic management regulations (in Chinese)*. Civil Aviation Administration of China.

[2] CAAC (2018). 2017 civil aviation industry development statistics bulletin (in chinese). http://www.mot.gov.cn/tongjishuju/minhang/201806/P020180621341239857728.pdf.

[FAA] FAA. Automatic dependent surveillance-broadcast (ads-b). https://www.faa.gov/nextgen/programs/adsb/.

[4] Wiegmann, D. A. and Shappell, S. A. (2017). *A human error approach to aviation accident analysis: The human factors analysis and classification system*. Routledge.