Informatics Large Practical

Coursework 2 Report

Felicity Wang
s1912558

December 10, 2021

Content

1 Software architecture description

1.1 Outline

This section provides a description of the software architecture of my application and conveying their importance.

1.2 Requirements

The goal of the project was to develop a drone delivery services, within a confinement area on a map and avoid the no-fly zone area. Take reading of order details of provided date, fly to shops to pick up items and deliver to users within 1500 moves. If the drone was not able to deliver all orders within 1500 moves, it maximise the total price of delivery by choosing path using less moves, and fly back to where it started before 1500 moves used up.

1.3 Classes

1.3.1 WordsLocation

This is the Json structure for file details.json saved in WhatThreeWords[1] folder location. The structure file here is for further information loading, we need to convert the WhatThreeWords locations into pair of longitude and latitude.

1.3.2 MenusDetails

This is the Json structure for file menus.json. the structure file here is for further information loading, we need to get cost and WhatThreeWords location details by given item name from this file.

1.3.3 WebServer

This class is for getting connection to the server and get information from server. In this project we have web server and derby database server need to connect. The methods in this class is for connection and load information by different requirements through different server port. For example, getOrderNoByDate is connect to derby server first, find the corresponding table by parameter order data and return order number.

1.3.4 LoadInfo

This class is for loading information that need to connect to the server, also prepare the generation of final GeoJson file. Methods in this class get call method in WebServer with URL to fetch information from files. Getter in this class allow other classes access the data. It also pre-process some data for further usage.

1.3.5 DeliveryDetails

This class is the pre-processing for final plan of drone route. It pre-process the move value to determine the sequence of deliver, load landmarks, decide which landmark if the closest form given point, determine whether a move is valid by checking if it pass the no-fly zone and if it stays in the confinement area.

1.3.6 DroneRoute

This class is for determine the sequence of delivering orders, and plan route for the drone to send as much order as possible, and send the order that has greatest value per move first. Add landmark when necessary. Getters in this class is used to pass information to write in database and write as file.

1.3.7 SaveFlight

This class is for writing information into tables and files. Get data from class DroneRoute and create new table to save information of flight and write details of deliveries and flight path into tables. Also write flight path into GeoJson file.

1.3.8 App

The main class to run the project. Fetch information from other class and receive parameters to run the program. It call the DroneRoute class first to plan the flight path and call SaveFlight class to save information of deliveries and flight into database tables and GeoJson file.

2 Drone control algorithm

2.1 Determine sequence of sending orders

Due to large demand of deliveries, the drone may not be able to send all orders within 1500 moves. To maximum the profit, the drone need to send the order with relatively high price and relatively short distance first. Method planRoute in class DroneRoute is for this purpose. It fetch a list of orders, get price and distance for every orders, then calculate the value per moves of each orders. It sort the orders with the greatest value per moves first and the smallest value per moves as last.

2.2 Avoiding no-fly zone and leaving the confinement area

Restriction of flight in this project is it cannot pass through the no-fly zone and must stay in the confinement area. Method validMove and isConfined is for these purposes.

It uses Line2D [2] to make a line for the movement from current position to the next position, and another line for no-fly zone. After creating two lines, use intersectsLine() in Line2D to determine whether the line of movement intersect with the line of no-fly zone. If yes then it cross the no-fly zone so it is not a valid move.

For confinement area, if the current position and next position stay inside the confinement area then the movement from current position to next position is valid. This checked by longitude and latitude of the position and confinement area.

2.3 Whole flight path

2.3.1 Flight can finish within 1500 moves

After sequence of deliver is determined, the drone start to plan the route from start point – shops – deliver to – next shops - … - back to start point.
In this process, method addLandmark() determine whether it is a valid move between each point. If not, it add the closest landmark among a current to next position movement.

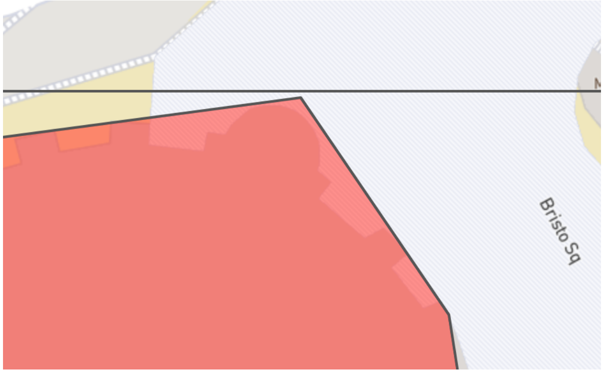
After finding all markers of movement in a flight, the drone starts to move by distance of 0.00015 degree. Start from the current position, travel in certain angle that can make the drone fly to the next marker of movement. When drone fly close to the shops or deliver to location, it hovers one more step for pick up and deliver. After finishing all deliveries, it moves from current position back to the start.
Move angle is the angle from current position to marker of movement, from 0 to 350. When the drone is hovering the angle is -999.
Definition of close to is Pythagorean distance between points less than 0.00015.

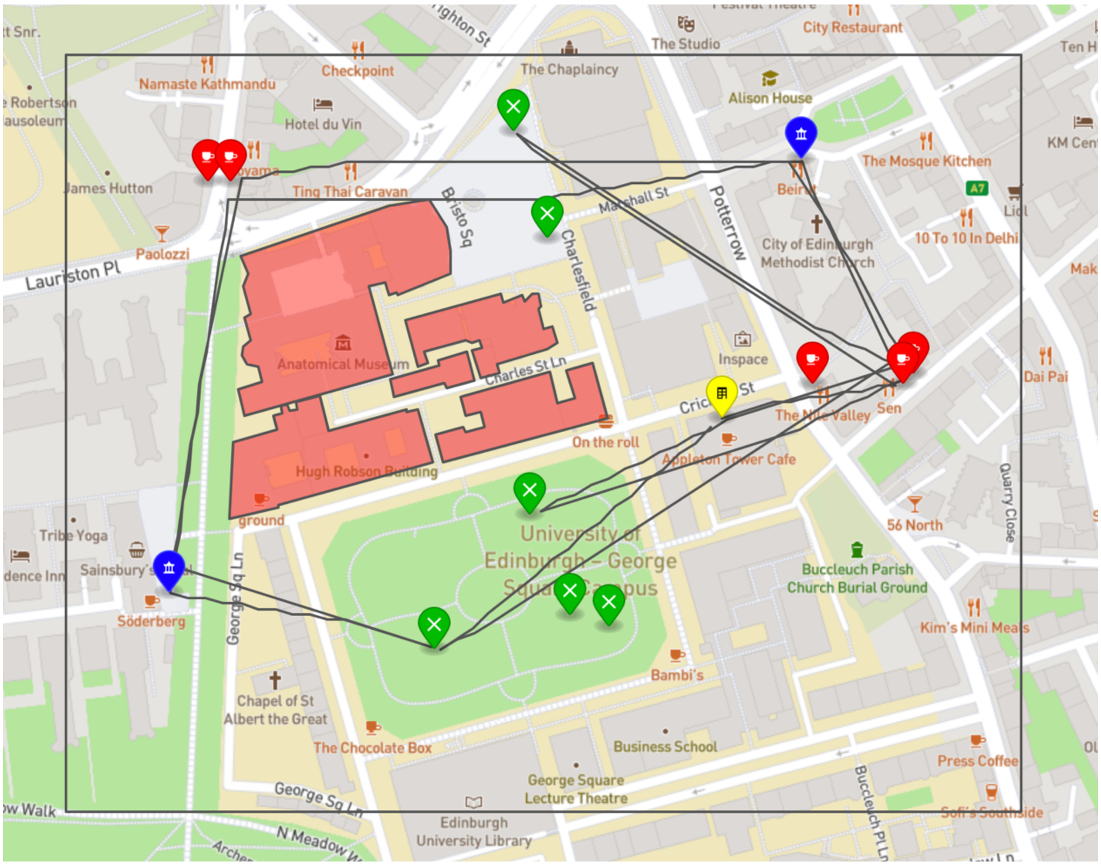
2.3.2 Flight cannot finish within 1500 moves

If the drone cannot finish all deliveries within 1500 moves, the way of calculating flight path need to be changed. From every step of movement, calculate whether the moves that drone already take plus moves back to Appleton Tower is within 1500 moves. Find the last point that the drone is able to fly back to Appleton Tower with in 1500 moves and plan the route among these markers of locations.

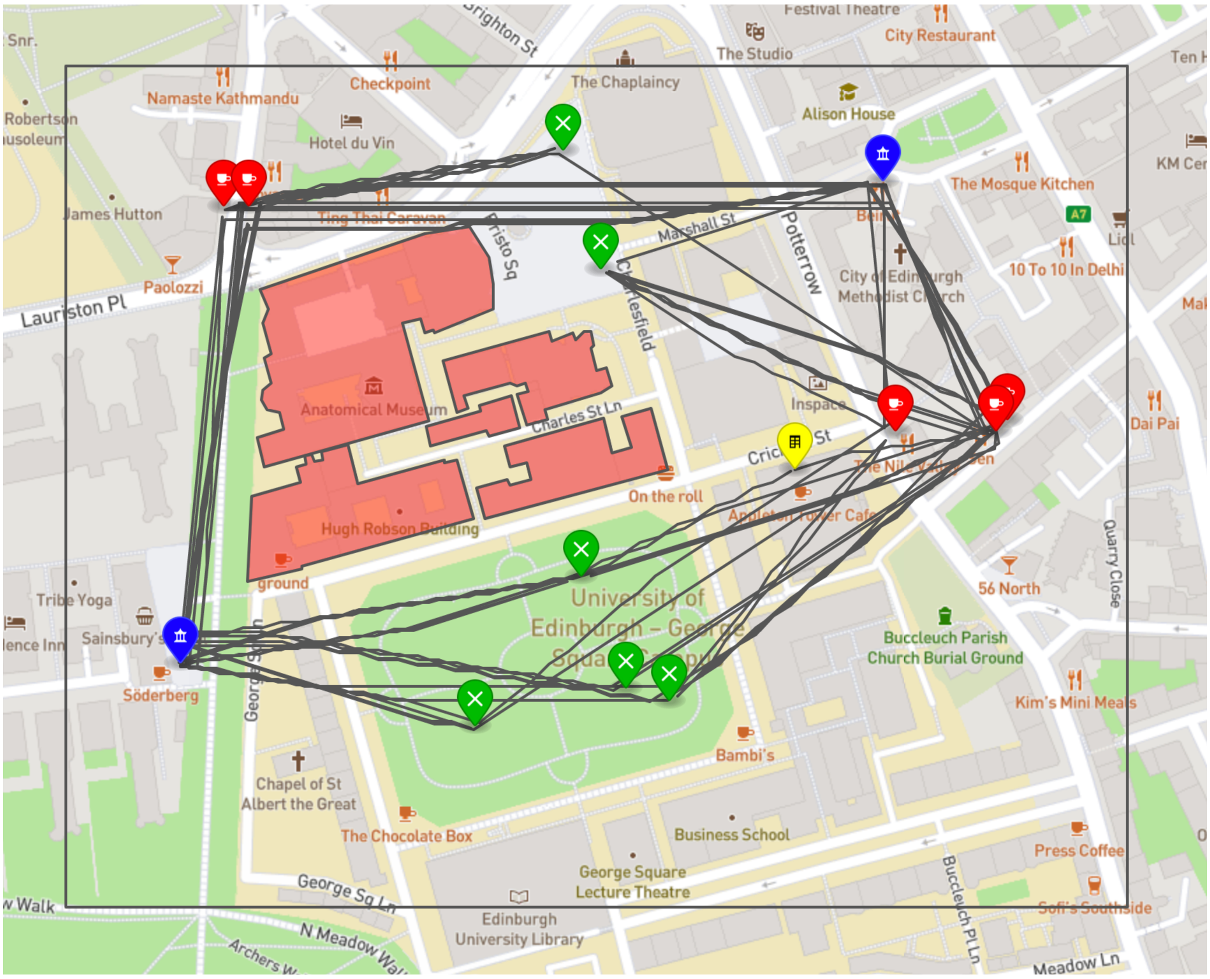
2.4 Example flight path

The first one an example that the drone finish the flight within 1500 moves. This is the flight path of 2022-01-01. We the see in the graph there is a flight path looks cross the no-fly zone, but I zoom the graph we can see it actually have a distance from no-fly zone.





The second example is the drone fail to deliver all orders within 1500 moves. This is the flight path of 2023.12.31, the drone fail to deliver last 2 orders and back to the Appleton Tower after a delivery.



2.5 Possible improvement

My project still have a lot of thing to be improve to increase the efficiency of the algorithm. In my project I only use provided landmark to avoid fly into the no-fly zone, this increase the moves of flight. There should be a solution that the drone can use shorter distance like fly close to the boundaries of no-fly zone.

Bibliography

[1] https://what3words.com/

[2] Line2D Intersection: https://stackoverflow.com/questions/24645064/how-to-check-if-path2d-intersect-with-line