Informatics Large Practical
Coursework 2 Report

Chengyu Kang

s1998909

December , 2020

Chapter 1

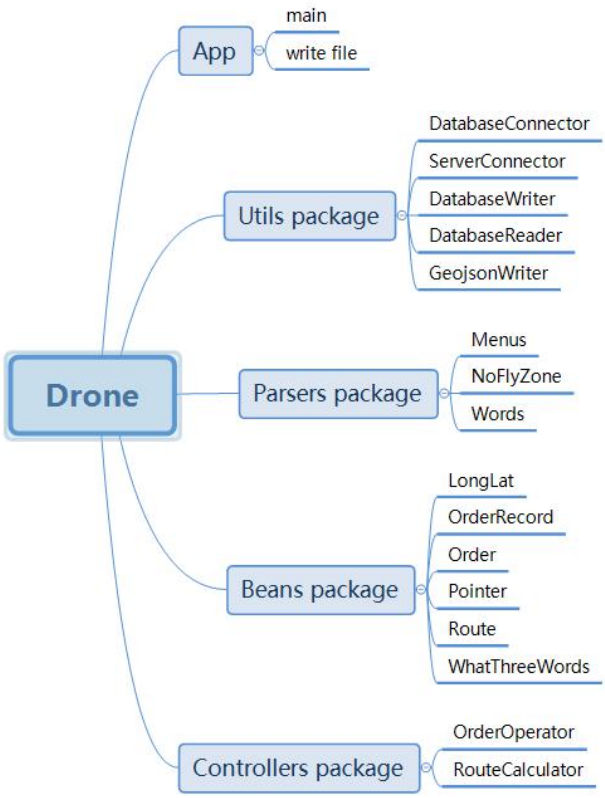
Software Architecture Description

1. overview

This section contains the framework of this project, and the purpose of creating them.

2. framework of the project

This project contains five main parts, those are App, utils, parsers, beans and the functional ones. **App** contains the main function of the whole project, **utils** provides some classes that have reusable server/database connectors, **parsers** parse the data from server or database, **beans** are entity classes that need to be used, and **controllers** contain main algorithm we use for calculating the drone flying path and sorting the orders.(figure1)



(figure1)

3. App

Main function is in the App class, it is responsible for running the whole drone delivery programme, the main function calls some other methods from the other classes, it first calls methods from classes in **utils** to get connection to database/server to get connection to them , then use classes in **parsers** to receive the data from them. Secondly it stores the data by creating entity classes in **beans**, which will be operated by the methods in the **controller** package. In the last, the main function will create tables in database to store the data of the whole procedure of the drone's flying, and will write the calculated fly path into the geojson file using the static write file function in main.

4. Utils package

This package has some classes which mainly read and write data with the database and server.

4.1 DatabaseConnector

The DatabaseConnector class is a connector to the database, which is reusable. It uses the given port and machine to get connection and statement from the database.

4.2 ServerConnector

The ServerConnector class is a connector to the server, which is reusable. It creates HTTP client and get response from the server.

4.3 DatabaseWriter

This class have some functions to write files in the database, to record the orders and the fly path of that day in the database.

4.4 DatabaseReader

This class provides functions to read data in the database, and get the result set of the target table.

4.5 GeojsonWriter

Geojsonwriter is a class that can transfer fly path from Longlat to a geojson file.

5. Parsers package

The parsers package contains some classes that will get data and parse it to useful data that we need.

5.1 Menus

Menus class is helpful when researching data from the menus data,it gets data from the server by the ServerConnector, and then it parses the data to a map,so either

shops or price related to the items can be searched by using Menus class.

5.2 NoFlyZone

The class NoFlyZone gets connection to the server by the ServerConnector, and it is used to get the no fly zone in the server as a feature collection.

5.3 Words

Words is responsible for parse a whatThreeWords string to a coordinate. It gets the whatThreeWords string and then search its related coordinate in the server.

6. Beans package

beans are entity classes, they will be used to simulate different objects in the project.

6.1 LongLat

Longlat is a coordinate which represents the longitude and latitude of the current position, it is used for marking the exact location of the drone during the flight. It also contains some functions to judge whether it is in our confined area or no fly zone.

6.2 OrderRecord

OrderRecord gets the data from the table “Orders” in the database, an arraylist of OrderRecord is the total order record of the given date from the customers.it’s a receiver class.

6.3 Order

the class Order is an object that is created to store the data from the OrderRecord, and also it have some variables that records the order's start and end location, it also stores the moves the drone needs to make in the is order. If there is one order that have two shops to go to, two orders will be created with different start location.Order objects will give useful information when sorting which is our next order to deliver. It's the key object in this whole project.

6.4 Pointer

Pointer as its name, it has its own coordinate, and it also has parent and son pointer as its nodes. Pointers will store the data in each coordinate the drone has been to. It helps to find a better route during the finding route procedure. At last, Pointers will form a one direction linked list to store the data during the flight of one route,including its moves, coordinate and total distance.

6.5 Route

Route has a last pointer of the linked list ,which can be traced back to a whole route, it stores the moves the drone needs to spend as well.It represent the fly path of the drone.

6.6 WhatThreeWords

WhatThreeWords stores what three words information from the server, it's a receiver class.

7. Controller package

Controller package has the important algorithms that sorts the more valuable orders and calculates the appropriate shortest route using A* algorithm.

7.1 OrderOperator

The OrderOperator have some functions inside which can operate the Orders. This class provides functions that can get the most valuable order of the current remaining ones by sorting out the one with highest sampled average percentage monetary value. It also contains functions that help transfer OrderRecord to Order and remove Order.

7.2 RouteCalculator

RouteCalculator provides function that can get the route from one coordinate to another using A* algorithm, without entering the no fly zone. It also have functions that help trace back the route. It's the core algorithm of calculating route in this project.

8. Third party objects

Mapbox Java SDK GeoJSON - It is used for handling and creating GeoJSON features.
Gson API - This is used to parse the JSON to Java classes.

Chapter 2
Drone control algorithm
1. Avoiding No fly Zones and unconfined area

To keep the drone fly in the legal areas to avoid the no fly zone, i wrote a algorithm that detects if the current move intersect with any boundary of the no fly zones.

So first draw a line from the start coordinate with the end coordinate, it's our current move line. Then iterate through all the buildings, since the buildings are also made up of different points, use another for loop draw a line between each points of the current building, detect if the flights move line intersects with the buildings' boundary line.

Here is the pseudo code:

```
Boolean isInNoFlyZone(start,end,noflyzone):
1.  Draw current move line
2.  For(each buildings of the noflyzone){
3.      For(all the points of the building){
4.          If(current line interact with the boundary line){
5.              Return true;
6.          }
7.      }
8.   }
9.   Return false;
```

2. Finding the nearest route from one coordinate to another

The algorithm I used here is very similar to the well known A* algorithm.But there are still some difference in details.

First of all, there are three lists contains the pointers , they are TotalPointers, waitingPointers and discoveredPointers.

Total pointers stores all the pointers that has appeared in waiting and discovered pointers. Waiting pointers are pointers that haven't been diffused while discovered pointers are those have been diffused.

Each time , we let the drone start from the pointer which have the nearest distance of starting distance add with ending straight distance. This means we will always start from the best pointer of the waiting list neglecting the no fly zone.

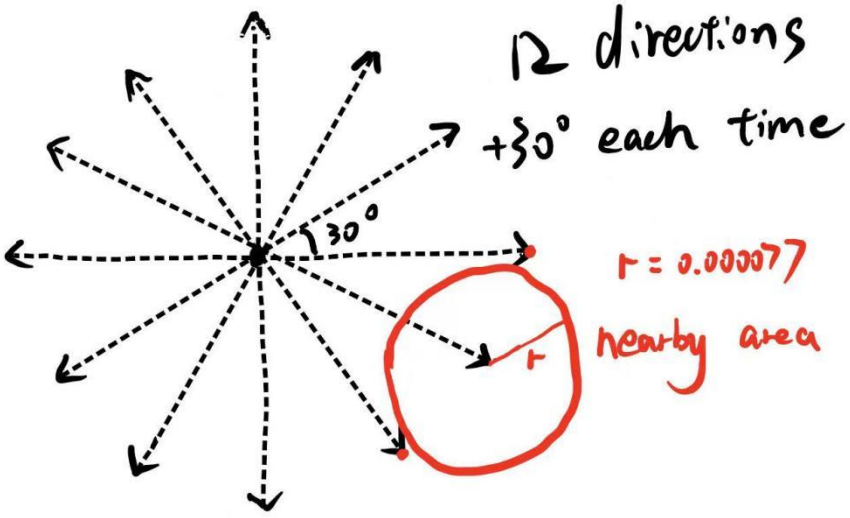
Then we diffuse from the current pointer in 12 directions from 0 to 360 at an

interval of 30 degrees.(This is because we considering the running time of the project, sacrifice some accuracy of the route). Each time we diffuse the point, we need to make sure it is not in the no fly zone or outside the confined areas. Also, we check if there are any nearby points we have been to before, if there are, compare which is a better pointer, add the new pointer to the waiting list when it is a better pointer or it has no nearby pointers.

The way to detect the nearby points is get the distance between two coordinates, if the distance is less than 0.000077, they are nearby pointers. The reason of choosing 0.000077 as our parameter is that the distance between the nearest diffused pointers is 0.0000773 , so we set a number that won't affect the nearby pointers and can cover as much area as possible to save our running time.

In the end this algorithm will give us a route that have the shortest distance to the destination avoiding the no fly zone meanwhile.

The figure below shows the diffusion from one point and the nearby area is in red circle.



(figure3: 12 direction diffusion and nearby area)

The pseudo code of this algorithm in finding nearest route is given as below:

```
Route getRouteMove(start,end,noflyzone):
1.  TotalPointers              // all the pointers
2.  WaitingPointers            // pointers waiting to be diffused
3.  discoveredPointers         // pointers has been diffused
4.  While(true) {
5.      Current = shortest(waiting pointers)    // get the pointer with the shortest total distance
6.      Remove current from waitingPointers
7.      Add current to DiscoveredPointers
8.      if (current is close to end){                  // route founded
9.          Break;
10.     }
11.     For(12 angles each 30 degrees){          // diffuse from current pointer
12.       Get the newPointer
13.       newPointer.setStartDis((current move+1)*step_length)
14.       newPointer.setEndDis(straight line distance to the end)
15.       Set total distance of new pointer
16.       newPointer.setPreviousPointer(current)
17.       If (the drone is not in illegal areas ){
18.             For(totalpointers){
19.                   If( new pointer is neighbour pointer the previous one){
20.                       Set break flag if new pointer has longer total distance
21.                   }
22.             }
23.       }
24.       If(there were better pointers nearby){
25.           continue
26.       }else{
27.           Add the new pointer to the total pointers
28.           Add the new pointer to the total pointers
29.       }
30.   }
31. }
32. Hover in the last pointer
33. Return the last pointer as a route
```

3. Deliver the order with the highest sampled average percentage monetary value and detect back moves

In this part i used the greedy algorithm. Before delivering the next order, calculate

the total delivery distance and the back distance, if the drone can fly back after the delivery, calculate the cost/totalDistance as the value of the order, and pick the one with the highest value in the end as our next order. If the drone selected null,it means that the drone is not able to deliver any orders and fly back, the drone will fly back to the appleton tower at this time.

Here is the pseudo code:

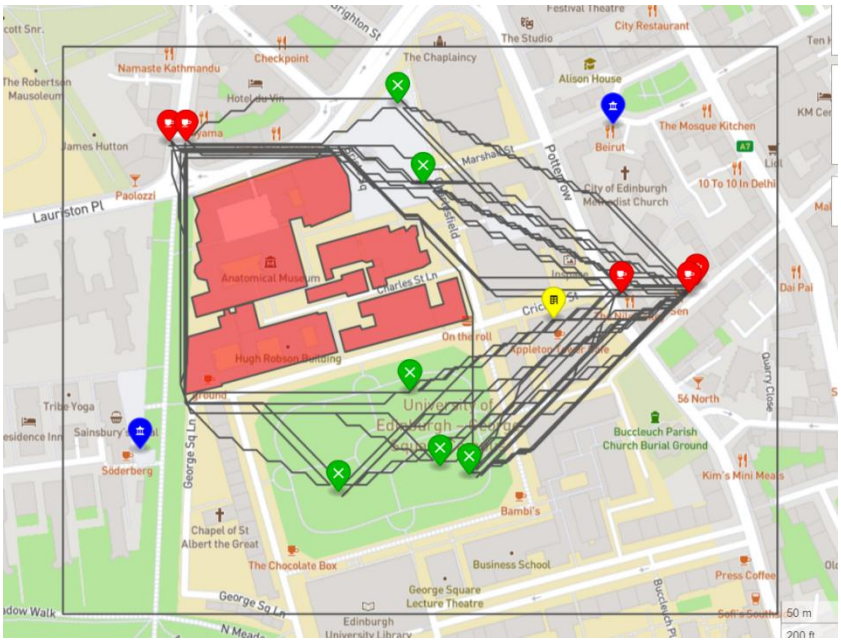
```
Order getNextOrder(orderlist, currentLocation, moves):
1.  For(orders in the orderlist){
2.      Get the order cost
3.      Calculate moves from current to this order
4.      Calculate the delivery moves
5.      Calculate the back moves
6.      Order.setTotalMove(toOrderMove+deliveryMove)
7.      If(left move of the drone< totalMove+backmove){
8.          continue
9.      }
10.     Get money/moves of this order
11.     Order = max(thisOrderValue order, maxValue order)
12.     maxValue = max(thisOrderValue,maxVale)
13. }
14. Return the route of the order with the highest value
```

4. The procedure of the algorithms

When the drone is delivering, it first uses the greedy algorithm(3) to get the order with highest value, then it will use the A* algorithm(2) to calculate the route during the delivering procedure, when calculating routes, algorithm(1) is used to avoid flying into illegal areas. When the drone is not able to fly back after one order, this order will not be considered in the greedy algorithm(3), if all waiting orders are unavailable or all orders were delivered, the drone will fly back to the appleton tower.

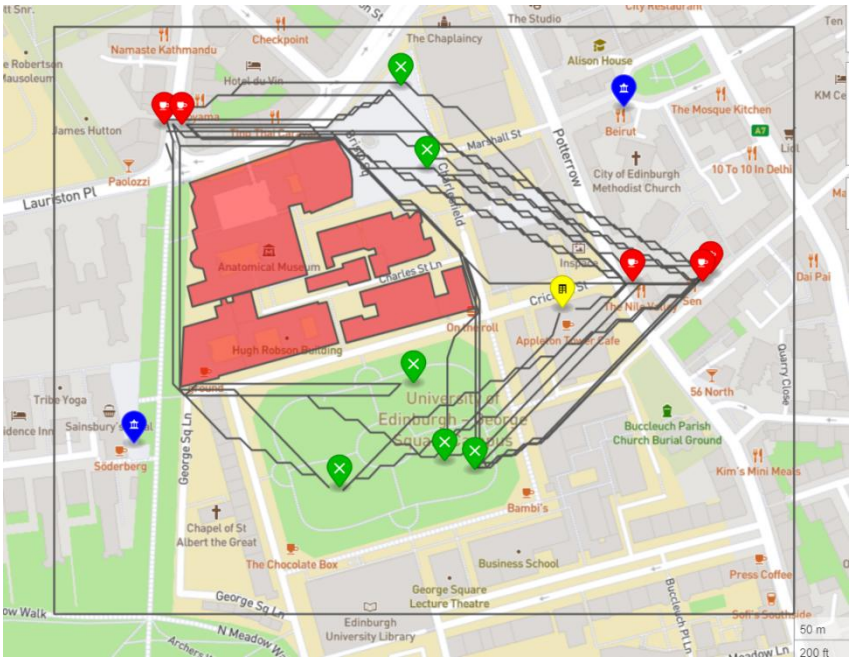
5. The example

This is the flight path of a random date orders, this is in 15/11/2023, the drone delivered 26 orders and there are 71 moves remaining after the whole flight, all the orders are delivered in order of current highest value.And the no fly zones are avoided.



(figure 4 : 15/11/2023 drone flight path)

This is in 03/10/2022, the drone delivered 13 orders and there are 709 moves remaining after the whole flight, all the orders are delivered in order of current highest value.And the no fly zones are avoided.



(figure 5: 03/10/2022 drone flight path)

References

[1].A-star algorithm - Cornell University Computational Optimization Open Textbook - Optimization Wiki

[2]Chen, Ge, et al. “Research on Ship Meteorological Route Based on A-Star Algorithm.” Mathematical Problems in Engineering, vol. 2021, 2021, pp. 1–8., https://doi.org/10.1155/2021/998973

[3].https://javadoc.io/doc/com.google.code.gson/gson/latest/com.google.