**Informatics Large Practical Report**

**Section 1: Software Architecture Description**

**Section 2: Drone Control Algorithm**

The Algorithm employed by the drone is a greedy approach.

Firstly all the valid orders made on a certain day are sorted, in ascending order, by distance from the starting position of Appleton Tower to the restaurant the pizzas are to be picked up from. This means the orders that are initially closest and would require fewer drone moves are prioritised and delivered first, to greedily increase the number of orders made by the drone within the limited number of moves.

Then the algorithm iterates through all of the valid orders, in this sorted arrangement. For each order, the drone generates the flightpath, which it calculates by greedily choosing the legal move that will minimise the Euclidean distance between the drone’s position after the move and its goal of the restaurant, move by move. Then once the drone is close to the restaurant it hovers to pick up the order. Then after this, the path to the restaurant is reconstructed in reverse order, with the angle of the move reversed by adding 180 degrees modulo 360. This subsection of the flightpath is then added to the drone’s overall flightpath, and the order would be delivered on that day if the number of moves performed by the drone would not exceed 2000. Otherwise, that subsection of the flightpath is discarded, and not performed by the drone, the order is marked as valid but not delivered and the algorithm then moves onto the next order to see if it is possible to deliver.

The validity of a drone move is verified by solving the line-line intersection problem between the line formed by a drone move and each line segment along the perimeter of each of the no-fly zones. The solution to this problem is calculated by viewing the two line-segments in terms of Bezier parameters.[[1]](#footnote-1) Firstly we represent each line segment respectively as

Where and are the start and endpoints of a linestring, and

Where and are again the start and endpoints of a linestring.

We then calculate where along each line segment the intersection point of the two lines will be by:

We note that if the denominator of these two terms is 0 then these values will not exist, and so these two lines are parallel and will never intersect, therefore we check the value of the denominator is non-zero first. Then if the denominator is non-zero we calculate each of these values. If they are both between 0 and 1 inclusive then we note that the two line segments do intersect each other. Otherwise the two line segments will only intersect each other if at least one of them is extended past its start or end point.

1. https://en.wikipedia.org/wiki/Line%E2%80%93line\_intersection#Given\_two\_points\_on\_each\_line\_segment [↑](#footnote-ref-1)