Imperial College London

Department of Computing

Drone Delivery Network

Simulation on SpatialOS
Interim Report

Author:

Paul Balaji

Supervisor:

Prof. William Knottenbelt

Second Marker:

To. Be Decided

January 25, 2018

Contents

1 Introduction

Drone technology is becoming increasingly popular. Their agility and ability to be used remotely makes them ideal for a number of use cases in numerous industries such as film, law enforcement, emergency services, agriculture and commercial delivery[4].

Due to numerous advances in technology, drones are quickly advancing to the point where human input is no longer a necessity. This has led to many companies showing interest in integrating drones with their work in the coming years.

Although it may be an engineer's dream for a fully automated world, drones in particular are a harrowing reminder that there are real risks associated with them. There are already several incidents of drones crashing into planes and flying into areas they shouldn't, most notably near Heathrow airport[3]. All of this provides motivation to introduce some form of autonomous air traffic control system to navigate these drones to their respective destinations in a safe manner.

However, prior work has been done on the routing and navigation aspect of such a system. In order for drones to truly take over more aspects of our lives, we must look at how they can provide a tangible cost-benefit to specific use-cases. There is no doubt that simply removing the human element can save costs drastically, and there is none more exciting a scenario than with deliveries. For example, Amazon stands to increase their margins considerably if they can successfully pull off their Prime Air initiative[7].

1.1 Autonomous Systems

1.2 Making Money

2 Background

We first provide an insight into drones, and the considerations to account for when applying them in day to day life. Additionally we summarise prior work completed by Imperial students on an autonomous air traffic control system.

We then continue to discuss modern delivery networks and introduce a mechanism by which economic value is prioritised. Finally, we give details about Improbable's SpatialOS and the reasoning for using this platform for the drone simulation.

2.1 Drones

As we have introduced, drones stand to be a revolutionary part of our lives as we welcome the new, incoming era of automation. However, to be practical there are a few key concepts one must understand to ensure that they remain a help and not a harm or hinderence to mankind.

2.1.1 No Fly Zones

No Fly Zones (which we will abbreviate as NFZs) are geographical areas where a drone is not allowed to enter or fly at any altitude. Examples of these may include Hyde Park, Buckingham Palace, airports and military locations. Typically these are static obstacles that will always remain a NFZ, however we could also consider cases when they could be created dynamically.

Suppose there was an issue of national security, it would be beneficial for the security and intelligence services to set up a temporary NFZ around areas where they deem a risk so that they can conduct their own operations without worry of external parties interfering with the situation.

2.1.2 Manned Aviation

Manned Aviation is any form of airborne transport with humans on board. This would include passenger jets, private jets, helicopters and other miscallaneous vechicles. For simulation purposes, we can consider these to have straight paths from a start to an end because relative to the zig-zagging of drones - they are effectively straight.

It is paramount to avoid collisions with manned aviation because there is a high risk of human calamity, in addition to the bad press and financial costs associated with such an air traffic incident.

2.1.3 Other Drones

Naturally we would want to make sure that we avoid colliding into other drones as well. In a perfect system, a single air traffic controller would have totalitarian dominance over all drones that take to the skies - however this is not *Black Mirror* and we have to assume that there will always be drones that this system will not be able to control or predict.

Nonetheless, a system can ensure that it navigates drones under its control as best as it can, avoiding these external drones and other rogue drones that may be flying with the sole intention of causing problems to others.

2.1.4 Toll Zones

Toll Zones are geographical areas where a drone has to pay an extra fee to pass through at any altitude. It is a very similar idea to the one that led to Congestion Charges being applied to much of central London. By restricting certain areas only to those who are willing to pay the fee to use the airspace, it reduces the density of air traffic in a specific area. These charges could also be used as an incentive to reduce pollution in the toll zone, and the additional revenue generated by the toll fees could be used towards the drone-related systems in place within the zone.

2.2 Autonomous Air Traffic Control (AATC)

2.2.1 What is AATC?

During the Autumn Term of the 2016-17 academic year, several students undertook a group project in association with Microsoft and Altitude Angel to produce an autonomous air traffic control system. The goal of the project was to safely navigate drones from their start to their end goals, whilst avoiding obstacles and taking the shortest possible path to minimise battery use.

2.2.2 Technical Overview

AATC has a simple client-server architecture, where drones connect to the REST service to send their telemetry information and goal waypoints every second. The server sends back direction recommendations to navigate the drone to its destination.

Because it would have been too expensive to trial the system on actual drones, a test bench was also implemented that would simulate the drones polling the server and responding to its recommendations. This test bench produced a set of simulation data, that is then available to view on the AATC visualiser[1].

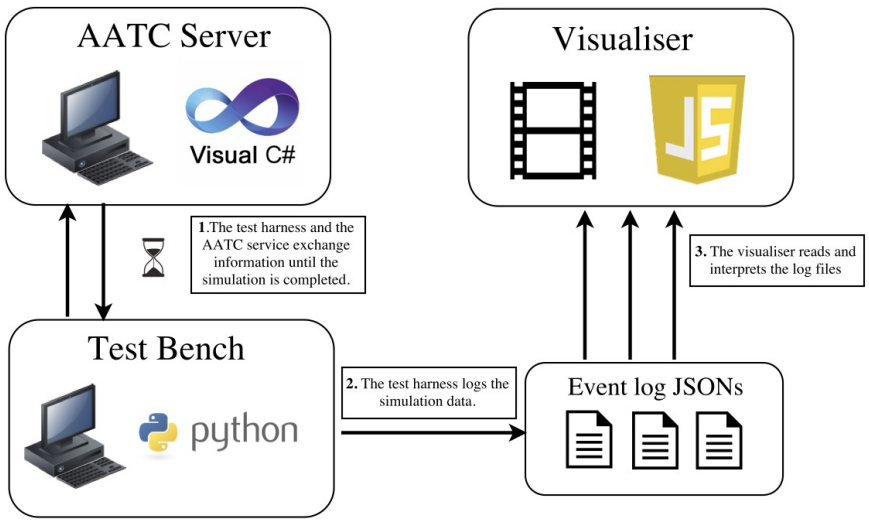

Figure 1: Technical Overview of AATC. [2]

The system was designed with three challenges in mind. The first challenge was to route drones from their origins to their destinations, and secondly, to ensure they avoided collisions with both static and dynamic obstacles - such as the ones mentioned in Section 2.1. The last challenge is to design the system in such a way that it would be able to scale to hundreds and thousands of drones.

To this end, a *Global Layer* was built to deal with static obstacles (such as NFZs) by calculating a path for the drone around NFZs before it begins its journey. The second problem was tackled with a *Reactive Layer* that handles non-static obstacles (such as manned aviation and other drones). This is the layer that would be providing the real-time updates to drones as they poll the server every second.

2.3 The Global Layer

The Global Layer holds the static representation of the world so that given a start and end, it can compute an optimal set of waypoints that a drone should follow - thereby dealing with AATC's path-finding problem.

2.3.1 Dijkstra's Algorithm

Dijkstra's algorithm is a popular algorithm to find the shortest paths between nodes in a graph. When using a co-ordinate grid system, each coordinate could represent a node in a graph and thus the algorithm can also be used to find the shortest path between a source and destination. Appendix A is an example implementation of Dijkstra in Python.

However, in the real world, we also have to consider the cost of computation and potentially make use of heuristics in order to return the shortest path given a limited amount of time. Especially in the drone case, we want to compute a good path as quick as possible in order to let the drone proceed along its way.

2.3.2 A* Algorithm

Enter, the A* algorithm. This algorithm is a generalisation of Dijkstra's algorithm that reduces the number of nodes explored during the search process by use of a heuristic - typically a minimum distance to the destination. A benefit over Dijkstra in particular is that it considers the distance already traveled into account, which aids the heuristic mechanism.

Naturally, by searching less nodes on a graph, less computation is performed and therefore A* has better performance than just using a pure form of Dijkstra's algorithm.

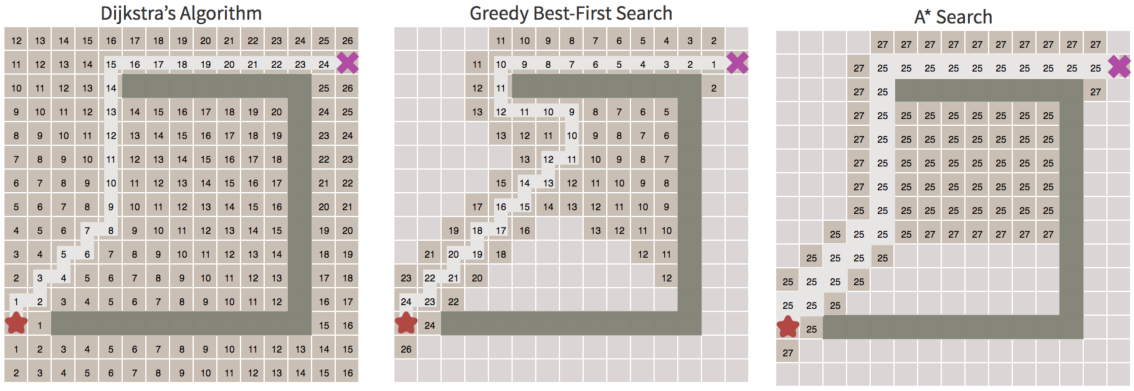

Figure 2: Comparing popular search algorithms. [2]

But for all these positive aspects, one must remember that our use for this algorithm is to compute paths in a real world, which is not necessarily split up into a nice, clean grid. So we turn our attention to a modification of the A* algorithm: Theta*.

2.3.3 Theta* Algorithm

The Theta* algorithm is an any-angle pathfinding algorithm based on the A* algorithm, which means that each jump from node to node in the returned path can be at any angle and not just up, down, left or right. This neat addition allows Theta* to be capable of finding near-optimal paths with a runtime comparable to A* [6].

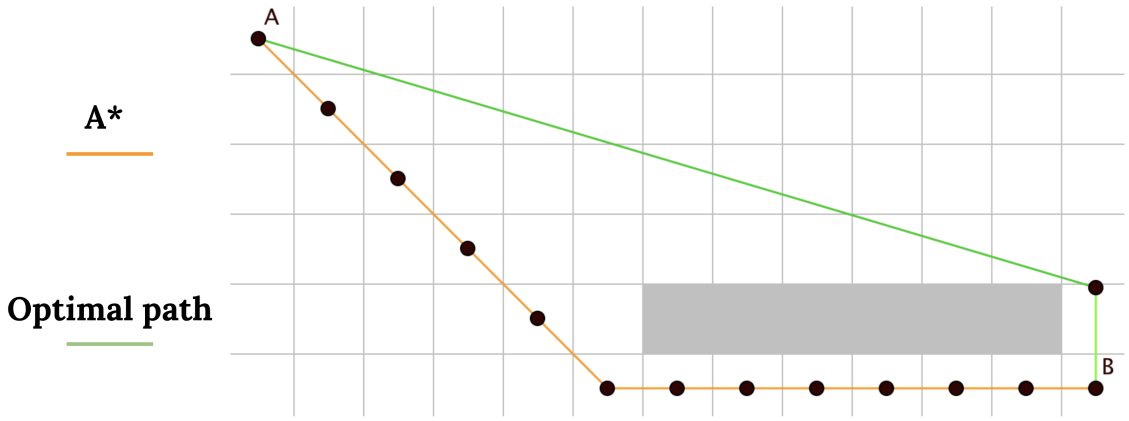

Figure 3: A* VS Optimal Path. [2]

Being able to get as short a path as possible is absolutely vital in the drone use-case because they have limited flying time. After all, their batteries can only last so long before they need to be recharged. Therefore, it is important to ensure drones travel as little distance as possible in order to maximise the number of times they can be used between charges.

2.4 The Reactive Layer

The Reactive layer considers any dynamic obstacles that the drone may face, such as manned aviation, other drones and potentially dynamic NFZs.

2.4.1 Where to take it next?

2.5 Delivery Networks

2.5.1 Planes

2.5.2 Trucks

2.5.3 Drones

2.6 Prioritising Economic Value

2.6.1 Quality of Service

2.6.2 Value Curve

2.6.3 skdbsa

2.7 SpatialOS

2.7.1 Unity SDK

2.7.2 Layered Simulation

2.7.3 Distributed Simulation

Example text 1

Example text 2

Example text 3

3 Project Plan

3.1 Phase 1: Porting Global and Reactive Layers to SpatialOS

3.2 Phase 2: Implementing a Scheduling Layer

3.3 Phase 3: Visualising the Economic Value

3.4 Stretch Goals

4 Evaluation Plan

References

[1] P. Balaji, D. Cattle, A. Janoscikova, G. Peycheva, J. Matas, and S. Wood. AATC Visualiser, 2017.

[2] P. Balaji, D. Cattle, A. Janoscikova, G. Peycheva, J. Matas, and S. Wood. Autonomous Air Traffic Control. Technical report, 2017.

[3] BBC News. Passenger jet approaching Heathrow in drone 'near-miss', 2017.

[4] R. Koontz. Drones to the Rescue, 2014.

[5] L. Root. Python implementation of Dijkstra's Algorithm.

[6] T. Uras and S. Koenig. An Empirical Comparison of Any-Angle Path-Planning Algorithms. *Aaai*, 2015.

[7] A. Welch. A cost-benefit analysis of Amazon Prime Air A Cost-Benefit Analysis of Amazon Prime Air. page 57, 2015.

Appendices

A Python Implementation of Dijkstra's Algorithm

```
def dijsktra(graph, initial):
  visited = {initial: 0}
  path = {}

  nodes = set(graph.nodes)

  while nodes:
    min_node = None
    for node in nodes:
      if node in visited:
        if min_node is None:
          min_node = node
        elif visited[node] < visited[min_node]:
          min_node = node

    if min_node is None:
      break

    nodes.remove(min_node)
    current_weight = visited[min_node]

    for edge in graph.edges[min_node]:
      weight = current_weight + graph.distance[(min_node, edge)]
      if edge not in visited or weight < visited[edge]:
        visited[edge] = weight
        path[edge] = min_node

  return visited, path
```

Listing 1: Example Implementation from GitHub. [5]