Distributed Path Tracking for an Aerial Network

Antonio Afonso
Dep. of Electrical and Computer Eng.
Fac. of Engineering, University of Porto
Porto, Portugal
up201302913@fe.up.pt

Joel Vaz
Dep. of Electrical and Computer Eng.
Fac. of Engineering, University of Porto
Porto, Portugal
ee11271@fe.up.pt

Ricardo Dinis
Dep. of Electrical and Computer Eng.
Fac. of Engineering, University of Porto
Porto, Portugal
up201708994@fe.up.pt

***Abstract*—In this work, a distributed control algorithm for surveillance on an area of interest using Unmanned Aerial Vehicles was implemented using a existing physical simulator. It was achieved using a PID controller on each UAV's thrust while taking into account the distances between the neighbouring UAV to compensate for unwanted changes in distance, as a means to keep the formation. The drone's movement follows an optimal route based on a sequence of user-defined number of waypoints. Upon reaching a waypoint, the UAV captures an image and sends it to a base station where a global frame is assembled.**

I. Introduction

Advances in Unmanned Aerial Vehicles (UAV), such as, better localisation, wireless communication interfaces, and processing power, are enabling a myriad of new applications. In this work, we studied and implemented an application where a central station defines an Area of Interest (AOI) to be surveyed. This area will be larger than what a camera from a single UAV can capture and, in order to overcome this limitation, it is proposed to have each UAV coordinate their movement between various assigned waypoints and capture a frame on each. By combining the frames captured on each waypoint, the image of the whole AOI (global frame) can be updated.

One of the key performance indicators of this system would be the rate at which the global frame is refreshed, and it is mostly dependent on the desirable AOI, number of UAVs, path optimisation and speed at which the UAVs travel. The other one would be the coverage, which is dependent on the amount of waypoints, camera FOV and the AOI's dimensions

Our work will be implemented on top of an existing physical simulator and API that allow us to control a number of UAVs and simulate their movement and communication, adding non-ideal characteristics, such as wind(which will influence the movement), packet loss and collision, which need to be taken into account when implementing the communication system [1].

II. Solution

Our implementation consists of a base station and a variable number of UAVs.

The user inputs in this base station the number of desired drones in the swarm, the number of waypoints, and the distance between them, which will be then sent to all drones in an initialisation packet. It is also where the frames are received and assembled into a global frame.

Since the global shape of the waypoint path is known, according to [1] , the UAVs themselves can generate the coordinates of the whole sequence of waypoints to visit, given the distance between waypoints and their number on a side of the square AOI

The current position of an UAV is periodically sent to the the UAV directly behind and ahead of them in the route and adjust their own thrust, which in turn changes it velocity, accordingly. Upon reaching each waypoint, they send a frame to the base station. As we didn't make use of a real(or simulated) camera, the UAV sends 1 MB of random data to the base station instead.

A. Communication

As the environment where this application would be implemented contains a fair risk of packet loss, it was decided to use UDP as the transport protocol, as it would be hard to maintain a stable TCP connection and it's overhead cost would be significant. To counter the risk of information loss, additional features, such as, packet fragmentation, acknowledgement, and re-sending, were built on top of the application layer as they were required.

It was decided that the communication of position the sender UAV and the two receivers should be done in 2 unicasts, as a UDP unicast doesn't create much overhead and allows for us to implement better delivery guarantees than with a multicast model.

For the frames transmission, since they have a relatively large size, it was necessary to fragment that data into various, smaller sized packets and assure their correct order(with a sequence number). In order to do this, a state machine approach was implemented. This state machine runs in a thread that waits for the program to signal a new frame with the SetFrame() function. In figure 1 we can observe the implementation of the mechanisms to send frames from the UAV side.

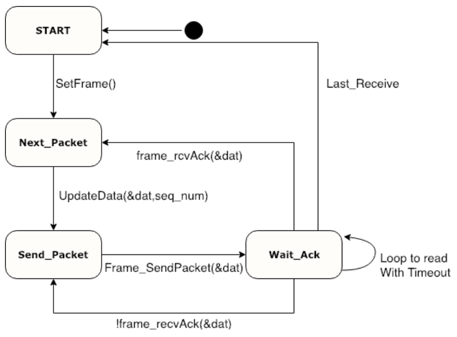


Fig. 1: Frame transmission mechanism.

In order to avoid collisions, it was attempted to implement the TDMA channel access method, however, after some technical issues with the API, it was decided to not include it in this project as it was not a priority.

B. Compute Waypoints

After receiving the initialisation packet from the base station the UAV generate one of two possible path ”styles”. The AOI is then decomposed in a list of waypoints to visit and each drone has to visit each one of them. In order to find the optimal path to take, this can be considered a subset of the Travelling Salesman Problem, meaning drones have to find the order in which to visit the waypoints so as to minimise the total distance of the path. The two path ”styles” calculated in [1] are used in our approach, one for an even number and another for an odd number of waypoints in the AOI.

Both ”styles” can be seen on figures 2 and 3. The odd path is slightly longer due to an existing diagonal. That isn’t considered in the presented algorithm as its influence is minimal.

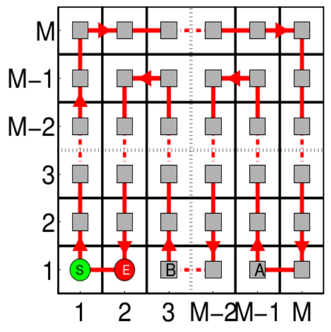


Fig. 2: Square AOI with even number of cells to visit.

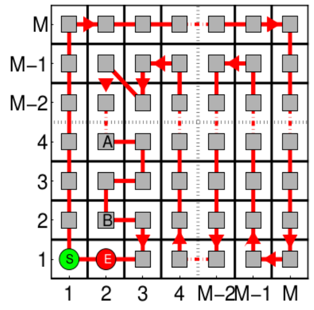


Fig. 3: Square AOI with odd number of cells to visit.

C. UAV Movement Control

The physical drone simulator presented in [1] tries to implement an UAV motion model in great detail, modelling several aspects of the drone’s behaviour in a real situation. Each UAV has a state of acceleration, velocity and position. Its acceleration is calculated according to Newton’s Second Law of motion, being the total force exerted on the UAV the sum of its propeller’s thrust and air drag. Since acceleration isn’t infinite, changes in velocity aren’t instantaneous, having the UAV 3 motion states: acceleration, cruising (in which it is at its maximum velocity) and braking.

On our solution the UAV thrust is being allocated with a PID controller, being its input the difference between the drone’s current position and the desired waypoint. To improve the rate at which the global frames are obtained, the UAVs are set to move at maximum speed by default.

In order to ensure that the distance between the front and back UAV are the same(so that the formation isn’t broken), each UAV is required to detect if it is closer to the front UAV than the back UAV and, if so, it shall reduce its velocity. If every UAV follows this rule, the network should be able to recover from any delays on any of the UAV’s movement.

Whenever an UAV reaches a waypoint, i.e. when it is within a tolerable distance from the waypoint, it pauses to send the newly captured frame and its new position, as can be seen in the diagram of figure 4.

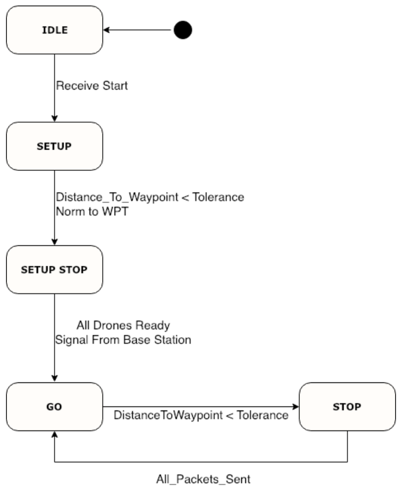


Fig. 4: Drone movement state diagram.

D. *Multi-Threading*

Multi-threading was important for this project as it allowed us to keep the code simple, due to the possibility of using blocking operations for sending and receiving packets through the network while being able to control the drone's movement independently. Each UAV used 6 threads:

- Sending data
- Receiving data
- Reading external sensor for GPS data
- Reading internal sensor for UAV information
- Sending frame state machine
- Movement Control

III. Results

A. *Frame packet loss*

By implementing an intentional failure to acknowledge mechanism on the base station(no acknowledge and wrong acknowledge), we were able to verify that the re-sending of packets was functional.

B. *Global Framerate*

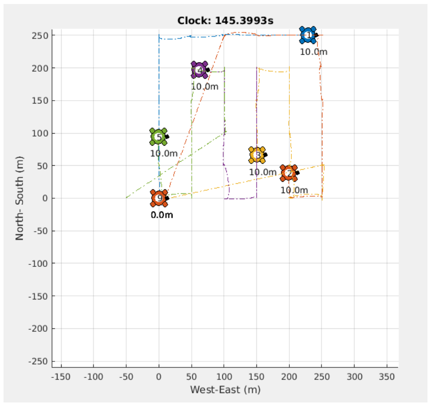
Using the configuration of 6x6 Waypoints with 50m in-between, we obtained the following framerates:

- 2 Drones: 0.0027 Hz
- 3 Drones: 0.0042 Hz
- 4 Drones: 0.0054 Hz

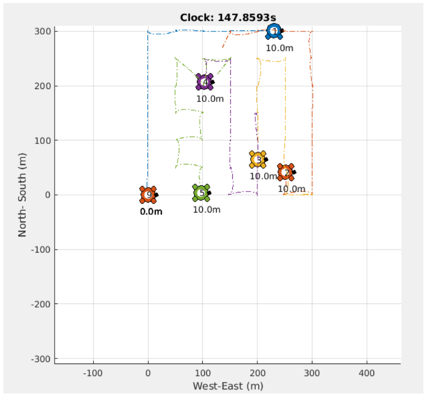
C. *Mapping*

In figures 5a and 5b, we can observe the movement of the drones. In figure 5a there are visible lines from around the starting point to other places, which represent the initialisation phase where the UAVs move to their intended initial position.

We can observe that the UAVs can correctly create the path with both an even and odd number of waypoints.



(a) Even number of cells to visit.



(b) Odd number of cells to visit.

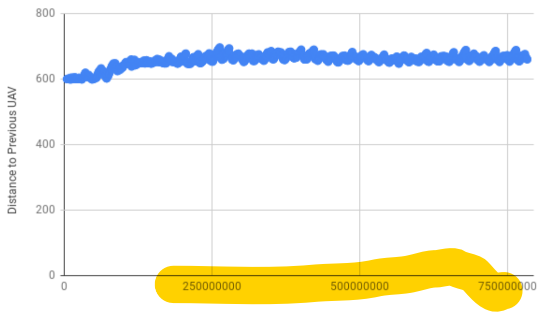
Fig. 5: Movement across AOI

D. *Distance Between UAVs*

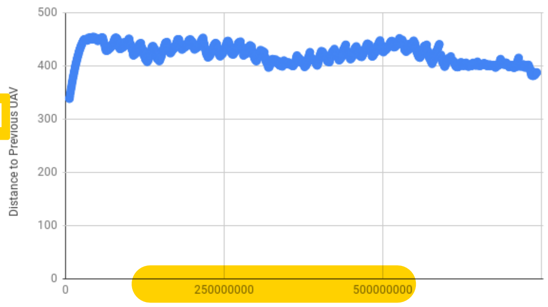
To observe the effect of the compensation algorithm, we can compare how the distance between 2 drones evolves over time.

In figure 6a we can observe that, after the previous UAV had a delay, the distance was never corrected back to its intended value(600), while in figure 6b, after multiple delays, the UAV will slow down and let the neighbour catch up, consequentially, the UAV ahead will do the same after noticing this UAVs distancing and so on.

In the annex, more results from different UAVs can be verified



(a) 3 UAVs system without compensation algorithm



(b) 4 UAVs system with compensation algorithm.

Fig. 6: Distance from an UAV to another neighbour UAV

IV. Analysis

When trying to evaluate the system's robustness, one common issue was erratic UAV movement resulting of inadequate PID parameters. Being this a dynamic application, where the user can, albeit not freely, choose the AOI to sweep, PID control of the rotors' thrust with fixed parameters isn't suitable. Parameters tuned to a certain waypoint distance do not translate to a smooth movement and stable frame rate when reducing that same distance since the UAV drone movement can uncontrollably oscillate. Tuning the parameters too conservatively as to have the application work under smaller waypoint distance ($< 10m$) can also result in a decreased performance when increasing the waypoint distance.

V. Work Distribution

- Joel Vaz: Presentation slides, Report
- Pedro Afonso: UAV movement control, Path building, Report, Data Gathering, Internal Software Architecture
- Ricardo Dinis: UAV communication, Report, Data Analysis, Internal Software Architecture

VI. Conclusion

We consider this project to be a moderate success as we were able to implement the travelling and path creation algorithm, and a working, simple to use, interface to send the frame to a base station. We also got the opportunity to apply some of this Curricular Unit's concepts to make a better project than we would have made otherwise.

At points, we had to optimise the amount of data we were sending to the network, as it would sometimes create great amounts of back off time from the packet collisions. We also had some difficulties getting used to the more terminal-based Linux developing environment, but we are glad that we were able to learn a lot from it.

A. *Possible Future Work*

In future work, it can be added the capability of capturing an image on the UAV, while using an image mapping software, such as Open Drone Map, to generate the full picture on the base station.

On the communication it could also be implemented a CRC check, to detect data corruption, a sliding window approach to reduce the time each UAV is stopped on each waypoint and multi-hop between them.

A last useful feature would be for the UAV to detect lower battery levels(from the internal sensor) and act upon it.

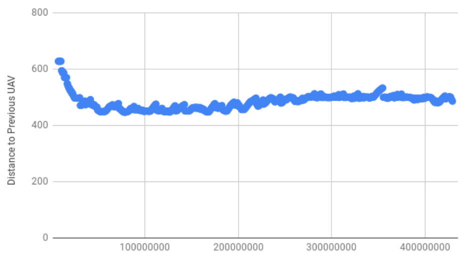
References

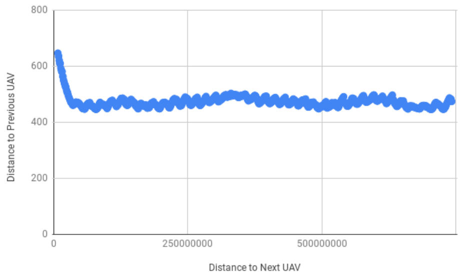
[1] L. R. Pinto, L. Oliveira, L. Almeida and A. Rowe, "Extendable Matrix-Camera using Aerial Networks", 2016.

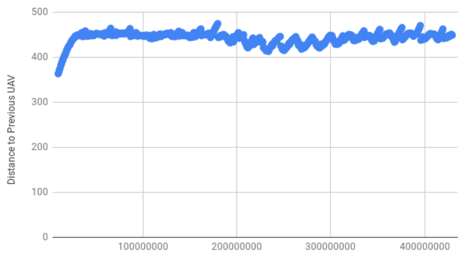
Annex

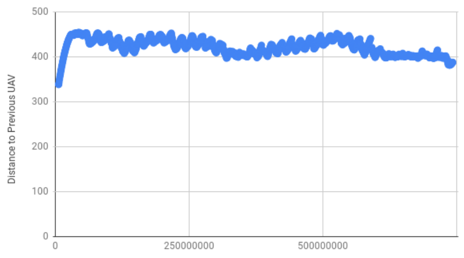
With Compensation

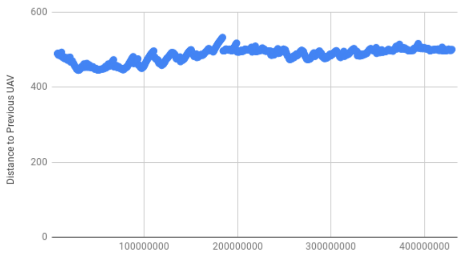
Without Compensation

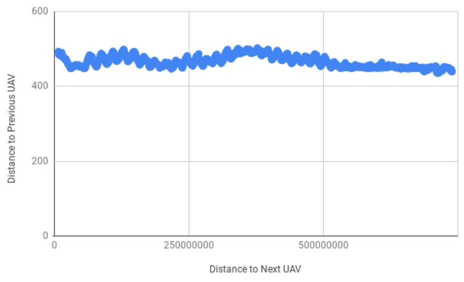

Distance to Previous UAV

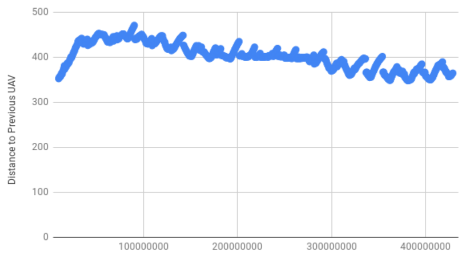

Distance to Previous UAV
Distance to Next UAV

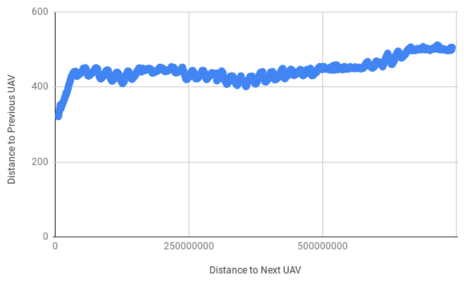

Distance to Previous UAV


Distance to Previous UAV


Distance to Previous UAV


Distance to Previous UAV
Distance to Next UAV


Distance to Previous UAV


Distance to Previous UAV
Distance to Next UAV