Posted on 11 May 2023 — The copyright holder is the author/funder. All rights reserved. No reuse without permission. — https://doi.org/10.22541/au.168379787.73912877/v1 — This a preprint and has not been peer reviewed. Data may be preliminary.

Decentralised Deconfliction of Aerial Robots in High Intensity Traffic Structures

Verdon Crann[1], Peyman Amiri[1], Samuel Knox[1], and William Crowther[1]

[1]The University of Manchester Faculty of Science and Engineering

May 11, 2023

Abstract

Projections for future air mobility envisage intensely utilised airspace that does not simply scale up from existing systems with centralised air traffic control. This paper considers the implementation and test of a software and hardware framework for decentralised control of aerial vehicles within intensely used airspace. Up to 10 rotary wing vehicles of maximum all up mass of 1 kg are flown in an outdoor volume with length scale of 100 m with GPS and WiFi connectivity. Flight control is implemented using a Pixhawk 4 flight controller running the PX4 firmware with guidance algorithms run on a separate onboard companion computer. Deconfliction is implemented using a simple elastic repulsion model with a guidance update rate of 10 Hz. Traffic structures are constructed from a path of directed waypoints and associated cross sectional geometry. Junctions are implemented when two paths converge into one or when one path diverges into two. Agents engage with structures through execution of flow, merge and swirl velocity rules. Calibration experiments showed that the worst case latency in agents sharing position information was of the order of 0 .5 s made up from delays due to finite guidance update rate, WiFi processing and centralised message processing. A choice of vehicle cruise speed of 2 m/s and conflict radius of 2 .5 m provided an acceptable compromise between experiment time efficiency (speed) and spatial efficiency (resolution) within the test volume. Results from recirculating junction experiments show that peak deconfliction activity occurs at the junction node, however biased distribution of agents within a corridor means the peak intensity is pushed ahead of the node. Use of meshed helical junction structures significantly reduces the intensity of conflict at the expense of reduced junction time efficiency.

Decentralised Deconfliction of Aerial Robots in High Intensity Traffic Structures

Verdon Crann 0000-0002-3725-726X, Peyman Amiri 0000-0002-8575-2978, Samuel Knox 0000-0002-2765-2813, and William Crowther 0000-0002-0613-9375

Abstract

Projections for future air mobility envisage intensely utilised airspace that does not simply scale up from existing systems with centralised air traffic control. This paper considers the implementation and test of a software and hardware framework for decentralised control of aerial vehicles within intensely used airspace. Up to 10 rotary wing vehicles of maximum all up mass of 1 kg are flown in an outdoor volume with length scale of 100 m with GPS and WiFi connectivity. Flight control is implemented using a Pixhawk 4 flight controller running the PX4 firmware with guidance algorithms run on a separate onboard companion computer. Deconfliction is implemented using a simple elastic repulsion model with a guidance update rate of 10 Hz. Traffic structures are constructed from a path of directed waypoints and associated cross sectional geometry. Junctions are implemented when two paths converge into one or when one path diverges into two. Agents engage with structures through execution of flow, merge and swirl velocity rules. Calibration experiments showed that the worst case latency in agents sharing position information was of the order of 0.5 s made up from delays due to finite guidance update rate, WiFi processing and centralised message processing. A choice of vehicle cruise speed of 2 m/s and conflict radius of 2.5 m provided an acceptable compromise between experiment time efficiency (speed) and spatial efficiency (resolution) within the test volume. Results from recirculating junction experiments show that peak deconfliction activity occurs at the junction node, however biased distribution of agents within a corridor means the peak intensity is pushed ahead of the node. Use of meshed helical junction structures significantly reduces the intensity of conflict at the expense of reduced junction time efficiency.

Keywords: Distributed Control, Aerial Robotics Framework, Unmanned Traffic Management.

Correspondence

Verdon Crann, University of Manchester, Manchester, UK
Email: verdon.crann@manchester.ac.uk
Funding: This work was part of the EPSRC CASCADE Program EP/R009953/1.

1 Introduction

Anticipated development of urban air mobility and aerial drone delivery services benefits from intensely utilised airspace that does not simply scale up from existing transport systems with centralised air traffic control (Airbus, 2018; Bauranov and Rakas, 2021; Federal Aviation Administration, 2020; Federal Aviation Administration and NASA, 2020). Current road transport systems use both centralised and decentralised management, e.g. traffic lights and delegated driver autonomy, respectively, and vehicles flow through predefined networks of various classes of roads connected via junctions. Similar concepts are proposed for 3D aerial traffic networks as in (Hoekstra et al., 2015; Jang et al., 2017; D. D. Nguyen et al., 2021; Quan and Li, 2020; Sunil et al., 2015; Tony et al., 2020). The present work addresses the problem of developing and testing algorithms for automated following of aerial traffic structures whilst maintaining safe separation between agents (deconfliction). The work is based on development of a practical outdoor test environment using commercial off the shelf drones operated within line of sight regulations. The aim is to progress the state of the art in deployment of laboratory-based aerial robotics techniques for multi-agent aerial traffic flow experiments. The following literature review is separated into three parts: the first considers frameworks for implementing a practical multi aerial robot experiment, the second Uncrewed Traffic Management (UTM) simulation, and the third latency in multi-agent communications.

A summary of existing frameworks developed to support practical aerial robotic applications is shown in Table 1. A framework is defined as a set of standards, libraries, and tools that integrates with one or more flight control hardware families. Telekyb (Grabe et al., 2013) provides a controller capable of implementing decentralised flight formations and supports trajectory planning, state estimation, and tracking. Mavwork (Mellado-Bataller et al., 2013) focuses on visual control of multiple micro aerial vehicles for indoor applications. Twirre (Van De Loosdrecht et al., 2014), was developed for vision-based autonomous flight of mini UAVs in both GPS-enabled and GPS-deprived environments. Paparazzi (Hattenberger et al., 2014) is an open-source drone hardware and software project for rotary and fixed-wing drones that supports popular autopilot firmware. In (Preiss et al., 2017), an indoor practical demonstration of a swarm of 49 Crazyflie 2.0 drones with a VICON positioning system was conducted. This work was extended with a SITL framework enabling simulation and integration with ROS and Gazebo (Silano and Iannelli, 2020). Aerostack (Molina et al., 2020; Sanchez-Lopez et al., 2017; Sanchez-Lopez et al., 2016) is a multi-layered framework for autonomous aerial robots that supports a wide range of applications and platforms and has been successfully used for outdoor applications.

There are several simulation frameworks that implement Uncrewed Traffic Management (UTM) rules that are relevant

Features	Aerostack (Sanchez-Lopez et al., 2016)	Paparazzi (Remes et al., 2013)	Telekyb (Grabe et al., 2013)	Twirre (Van De Loosdrecht et al., 2014)	Mavwork (Mellado-Bataller et al., 2013)
Compatibility with Pixhawk flight controllers	Yes	Yes	No	No	No
Demonstrated implementation of decentralised control	Yes	Yes	Yes	No	No
Compatibility with ROS	Yes	No	Yes	No	No
Built in SITL implementation	Yes	Yes	Yes	No	No
Does not require external motion capture system	Yes	Yes	No	Yes	Yes
Agent to agent communication supported	Yes	Yes	Yes	No	No
Maximum number of aerial robots demonstrated flying simultaneously	5	5	4	Not specified	Not specified
Open Source	Yes	Yes	Yes	Yes	Yes
Allows implementation of custom behaviours without modification of architecture	Yes	Yes	Yes	No	Yes

Tab. 1: Practical frameworks

to the present work. A simulation framework based on ROS and Gazebo was proposed in (Millan-Romera et al., 2019) and (Capitán et al., 2021) to develop in-flight deconfliction and control services, then implement and test them in customised configurations and scenarios, including use of automated threat management and conflict resolution. In (Carramiñana et al., 2021) and (Besada et al., 2022), an agent-based UTM simulator platform was presented to simulate the effects of availability of various UTM information sources and sensors on pre-flight and in-flight stages. A 2D agent-based Python simulation framework for low-altitude UTM systems including variable collision avoidance algorithms and useful definition of safety, capacity and efficiency metrics was introduced in (Ramee and Mavris, 2021). In (Zhao et al., 2019), a multi-agent air traffic and resource usage simulation framework was used to evaluate different air traffic management policies and obtain a relationship between policy, environment and resulting traffic patterns.

Also of relevance to the present work are studies investigating latency in multi-agent communications, which is a key driver of performance. The latency of a multi-agent robotic system was measured for up to 4 agents for different message sizes in (Berna-Koes et al., 2004). Communication back channels were proposed to decrease the latency. In (C. P. Nguyen and Flueck, 2011), a stochastic model which can be adjusted by system configuration to predict and simulate latency in multi-agent power grids was presented. In (Pasandideh et al., 2023) a systematic literature review on aerial robot networking was carried out which presents the limitations of existing flying ad hoc networks and identifies possible future work.

Whilst previous work identified the core components required for successful demonstration of distributed aerial control for UTM evaluation, the lack of an off-the-shelf solution for the specific research objectives required the development of a framework with bespoke elements. In particular, there was a need to implement custom on-board guidance algorithms and run efficient data gathering experiments with multiple drones.

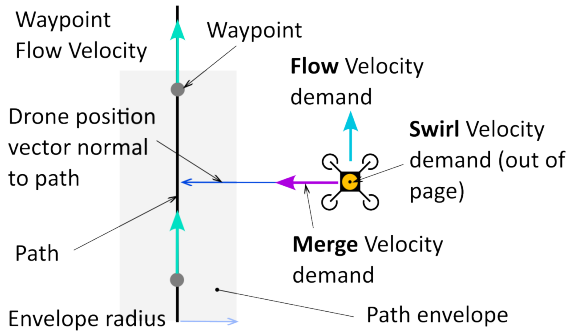

Fig. 1: Definition of flow, merge and swirl guidance velocities for agent path following

2 Traffic Management Implementation

An aerial routing network is comprised of paths (edges) and junctions (nodes). A path is defined both by its geometry and the guidance rules used to follow it. The geometry of a path is defined by a directed series of waypoints that forms a centreline. A cross sectional shape is swept along the centerline to form an envelope. Path guidance is provided by *flow* that governs the agent velocity parallel with the path centreline, *merge* which governs velocity normal to the centreline, and *swirl* that governs velocity normal to flow and merge, Figure 1.

Different types of paths are obtained by varying the topol-

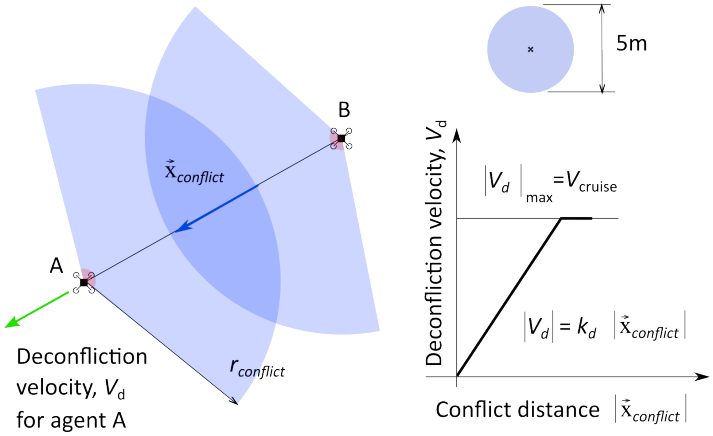

Fig. 2: Agent deconfliction model based on a virtual contact sphere.

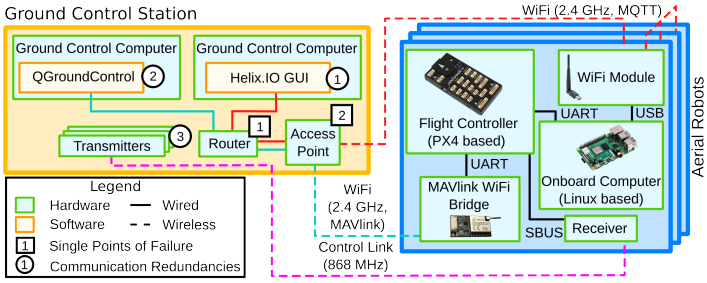

Fig. 3: System architecture diagram (Knox et al., 2022).

ogy of the path cross-section. A *line* has an envelope radius of zero. A *ribbon* has a cross-section formed by a line of finite length forming a 2D surface. A *cylinder* has a given envelope radius with agents able to occupy the volume anywhere within this radius. A *tube* is a special case of a ribbon which is wrapped around itself and the sides are connected to form the surface of a cylinder, with agents only able to occupy the cylindrical surface defined by the radius. Helical tubes are a special case of tubes in which agents have both axial and tangential velocity. Following from the definitions above, a line is consistent with common usage for lane, a ribbon is consistent with strip and a cylinder is consistent with corridor. There is no common name for tubular structures. A binary junction is formed when two paths combine into one path (a converging junction) or one path splits into two (diverging junction). Junctions are not reversible, i.e. the behaviour of a converging junction is not the same as that of a diverging junction with a simple sign change. Arbitrarily complex intersections can be formed by the assembly of multiple binary junctions.

Deconfliction is provided by a sphere around each agent defined by a conflict radius, Figure 2. This sphere behaves as a virtual contact surface that provides a repulsive velocity demand proportional to penetration with conflict spheres on other agents. The stiffness of the response is controlled by a deconfliction gain parameter. The maximum deconfliction demand is limited to vehicle cruise speed.

3 Methodology

3.1 Hardware Setup

The system architecture used for this work is shown in Figure 3.

3.1.1 Aerial Robot Hardware

Flight demonstration hardware was built around a custom five-inch frame racing quadcopter, Figure 4 and Figure 5. A weight breakdown for the vehicle is shown in Figure 6. Power was provided by a 14.8 V 3700 mAh lithium polymer battery and propulsion by four 2206 2300 kv motors running 5 inch propellers with a 3.5 inch pitch. Typical flight time for a standard configuration was 10 minutes. The principal avionics system components were:

- **Onboard Computer**: Raspberry Pi 4B with 2 GB of RAM used to run high-level control software, receive mission commands from the GCS, and send commands to the flight controller through UART.
- **Flight Controller**: Pixhawk 4 running PX4 Firmware used to provide low-level control and interfaces with the Mission computer via UART.
- **USB WiFi Communication Module**: Dynamode 2.4 GHz USB WiFi adapter, used to communicate telemetry information between agents and receive commands from the ground control program. Connected to the onboard computer via USB.
- **GPS Module**: Ublox NEO-M8N, used to provide position, velocity and heading information to the Flight Controller.
- **MAVlink WiFi Bridge Module**: Adafruit Huzzah with MavESP8266 firmware, used as a secondary control link to connect the Flight Controller with QGroundControl.
- **868 MHz Receiver**: TBS Crossfire Nano RX, provides a reserve data link through which agents can be controlled manually in an emergency, or a kill command can be issued.

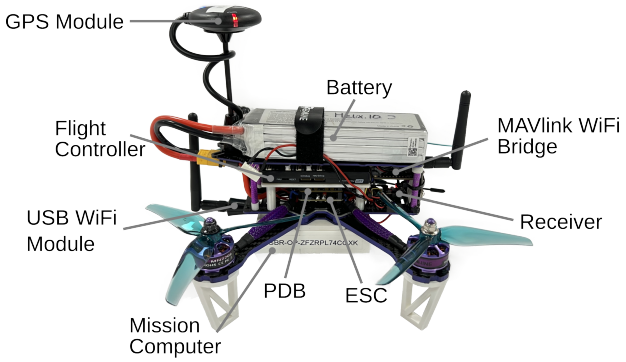

Fig. 4: Multirotor vehicle used for experimental work.

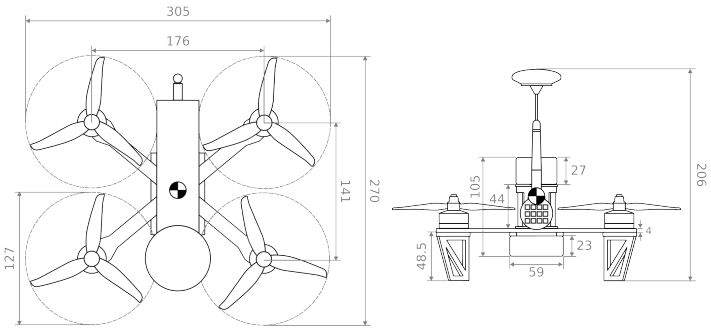

Fig. 5: Definition of main dimensions of the multirotor vehicle used for experimental work, all units in mm.

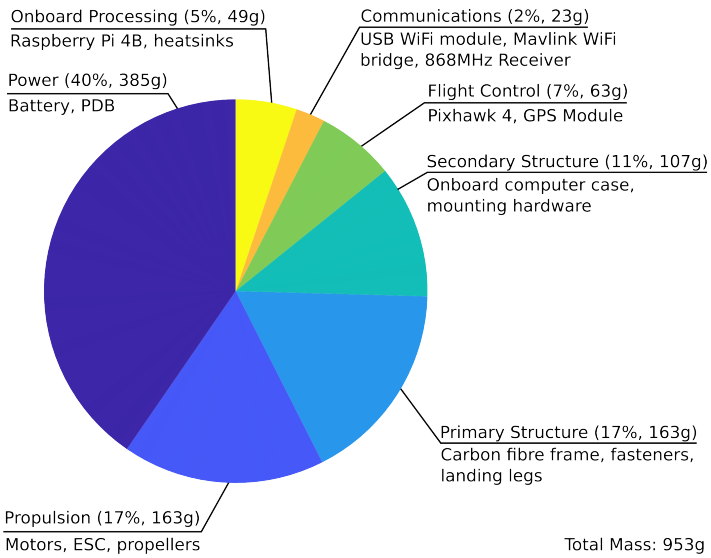

Fig. 6: Mass breakdown of the multirotor vehicle used for experimental work.

3.1.2 Ground Control Hardware

The Ground Control Station (GCS) hardware includes:

- **GCS Computer**: Runs the MQTT broker software, the experiment control program and QGroundControl
- **Router**: Manages the passing of packets between the access point and the GCS Computer, also handles assigning IP addresses to devices on the network.
- **Access point**: Used to improve the range of the router network using high-gain directional antennas.

3.2 Software Setup

The software architecture used for experimental work is shown in Figure 7. The software has three main components:

- **Experiment Monitoring and Control**: Runs on the GCS Computer. Shows agent status via a Graphical User Interface (GUI). Provides capability for sending control commands to agents individually or collectively. Example commands include take-off, land, hold, return to home, go to pre-start positions and start experiment.
- **Mission Management**: Runs on the agent Mission Computer. Implements high-level flight rules based on self position and position of other agents. Commands are sent to the Flight Controller using MAVSDK. Commands are internally generated or forwarded from the Monitoring and Control program. JSON files are used to define the geometry and parameters of each experiment. A flow chart for the code running on the mission computer is shown in Figure 8.
- **Communication**: Runs on the agent Mission Computer. An MQTT client provides a low latency publish and subscribe network protocol. Each agent publishes their telemetry to a topic which other agents can subscribe to. Brokerage is provided by software running on the Ground Control Computer.

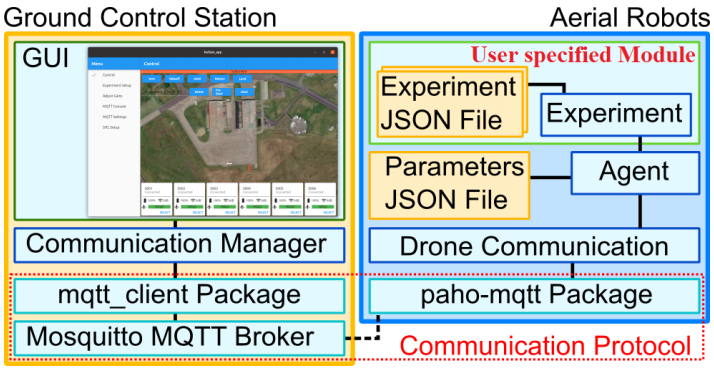

Fig. 7: Communication system and framework software (Knox et al., 2022).

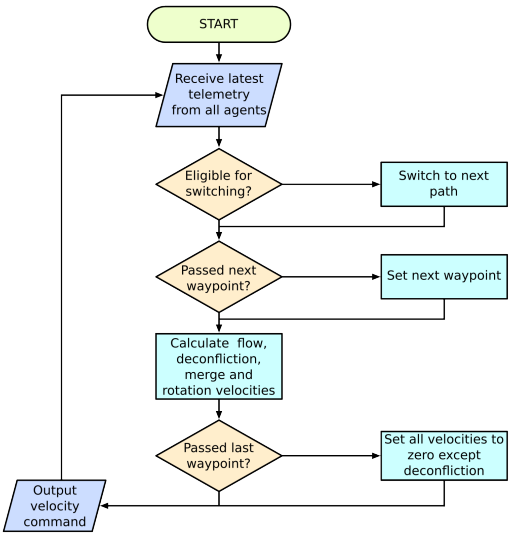

Fig. 8: Flow chart of operations performed on the onboard computer in one time step.

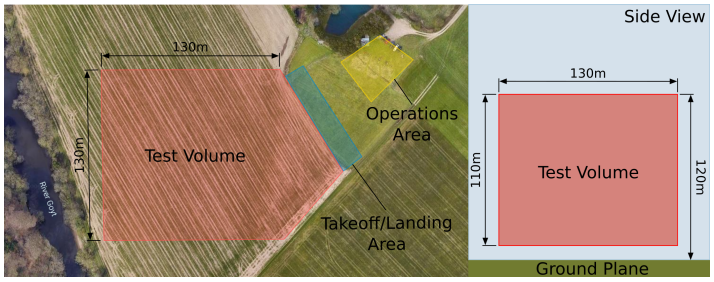

Fig. 9: Experiment test volume geometry and operational layout. Centre of test volume located at 53.408336, -2.124308.

A cross platform GUI application was developed in Dart to support field deployment of experiments. Users are presented with options for sending mission commands and experiment settings to agents. Data from each agent is displayed to show status and warnings. As a specific safety feature, a hold command can automatically be sent to all the agents if two or more agents get closer than an adjustable minimum spatial distance.

A Model In The Loop (MITL) simulation tool was developed to evaluate and verify guidance algorithms using idealised kinematic models for the agents with no disturbances and perfect sensing. This provided good initial estimates for the tuning parameters, which were then adapted and verified using Software In the Loop (SITL) experiments. SITL experiments were performed with the same set up as MITL, except using instances of the PX4 firmware simulated in a Gazebo model. This process was particularly useful for identifying collision risk, allowing operators to retune parameters accordingly. Parameters tuned using SITL were then implemented in hardware and tested in-flight. Finally, a visualisation tool was developed to plot agent trajectories from post-flight log data.

More details and information about the Framework have been provided in (Knox et al., 2022) by the authors of this work.

3.3 Experimental Method

Experiments were conducted at an outdoor flight test site within a working volume $130 \times 130 \times 110$ m box, Figure 9. Test volume was constrained by requirement for line of sight operations and a maximum ceiling of 120 m under UK CAA regulations. These dimensions also allowed adequate coverage using a single WiFi access point.

Choice of vehicle target cruise speed and conflict radius for the experiments was determined by consideration of the vehicle minimum stopping distance at maximum deceleration, including communication delays. Worst case is two vehicles approaching each other head on at cruise speed. A vehicle guidance update rate of 10 Hz was chosen based on suitable compromise between performance and stability. Vehicle maximum deceleration was set to the flight control system default of 3 m/s^2. The latency in sharing position information between drones is comprised of a WiFi transmit delay, the MQTT message processing time and a WiFi transmit delay, Figure 10. Experimental evaluation of the system loop-back time, Figure 11, showed that there is a baseline processing delay of around 0.05 s due to MQTT message parsing with an additional WiFi delay that increases proportional to the number of drones on the network, Figure 12. The worst case delay in reporting position information for two drones (out of a total of six) on a collision path was estimated based on a delay of 0.2 s, due to information being two guidance frames late and a worst case loop-back time also of 0.2 s, giving a total latency of 0.4 s. At a cruise speed of 2 ms^{-1}, the vehicle will travel 0.8 m during this period. Stopping distance from 2 m/s is around 1.4 m, verified by experiment, Figure 13., thus the total stopping required distance is around $1.4\ m + 0.8\ m = 2.2\ m$. The vehicle physical hardware is bounded within a sphere of radius approximately 0.15 m giving a minimum required deconfliction radius of $2.2\ m + 0.15\ m = 2.35\ m$. On this basis, a conservative value of 2.5 m was chosen for the conflict radius. This radius allows a maximum packing of 26 vehicles along the linear dimension of the test volume, which provides adequate spatial resolution for traffic flow experiments to be carried out. Choice of a faster cruise speed would have improved experiment productivity, but with adverse effect on spatial resolution. Choice of conflict radius here is conservative in that the head on collision case, whilst possible, is unlikely in practice because any slight offset in velocity alignment will cause drones to pass either side of each other.

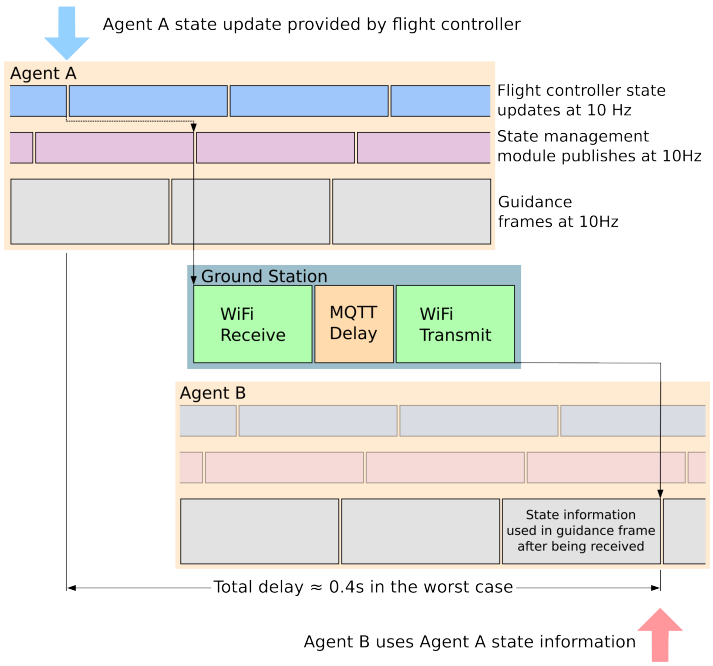

Fig. 10: Sequencing diagram illustrating the time delays in transmission of a message between two agents. Diagram drawn to scale with one guidance frame representing 0.1 s.

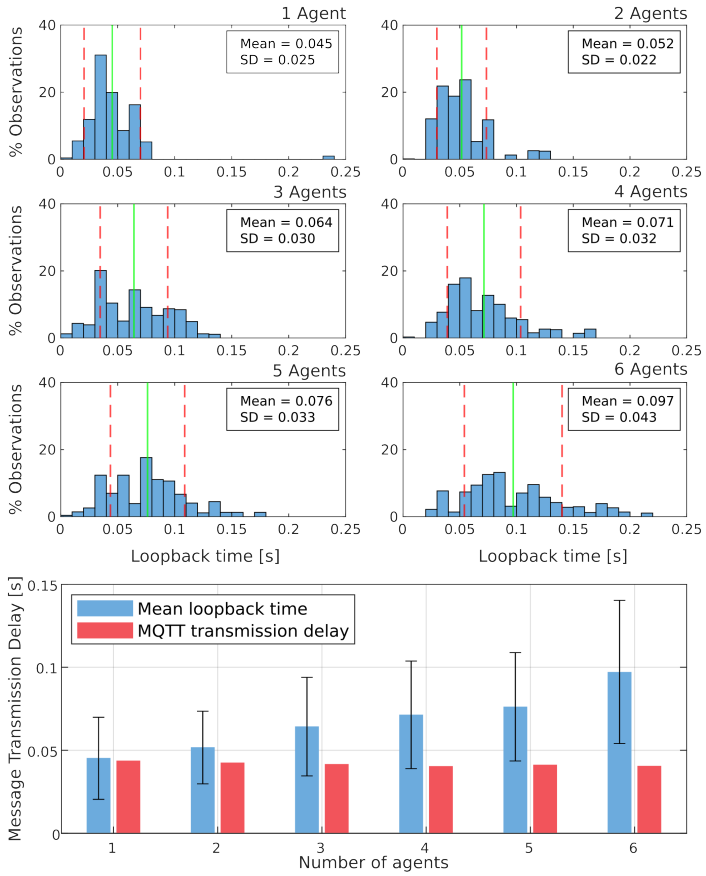

Fig. 12: Average loop-back time results for 1 to 6 agents. The green line on each sub-figure represents the mean, the dotted red line represents the standard deviation. The red bars in the bottom chart represent the results from a separate experiment where the message transmission delay of the MQTT broker was measured with different numbers of publishers and subscribers representing agents.

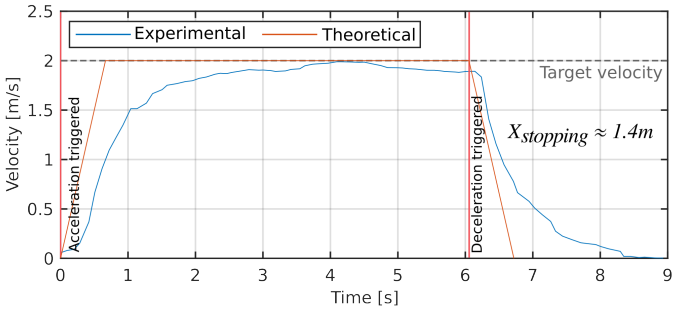

Fig. 13: Velocity-time plot for drone acceleration to cruise speed followed by deceleration to stop at max deceleration. Cruise speed $= 2\ m/s$.

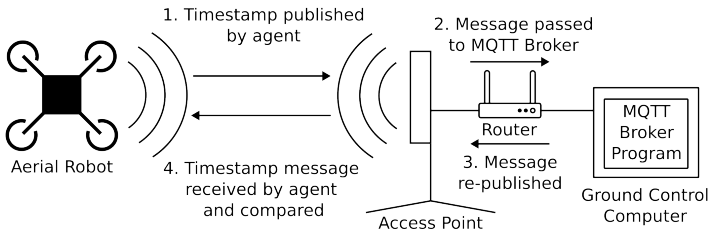

Fig. 11: Drone to base station communication path showing how loop-back time is measured.

The maximum practical number of drones in an experiment is limited by increasing latency and/or decreasing bandwidth per drone. Increasing latency increases the required conflict radius of each drone and hence reduces the physical number of drones that can exist conflict-free in the test volume. Bandwidth per drone has a minimum lower limit based on data packet size and the frame update rate. Figure 14 shows how the required bandwidth changes with number of drones assuming different guidance frame rates. Message rate is assumed to be quadratic with number of drones (all drones

Variable	Unit	Logging Rate
Time	s	10 Hz
Loopback Time	s	1 Hz
Current Waypoint	n/a	10 Hz
Position	m [NED]	10 Hz
Flow Velocity	m/s [NED]	10 Hz
Merge Velocity	m/s [NED]	10 Hz
Deconfliction Velocity	m/s [NED]	10 Hz
Swirl Velocity	m/s [NED]	10 Hz

Tab. 2: Data acquisition variables recorded onboard each agent.

share position information). For a guidance frame rate of 10 Hz and MQTT packet length of 50 bytes, the theoretical maximum number of drones based on bandwidth is around 116. In practice, the maximum would be less than this due to latency constraints at high drone count. Bandwidth constraints could be significantly reduced using a mesh network approach in which drone position was only transmitted to nearby agents.

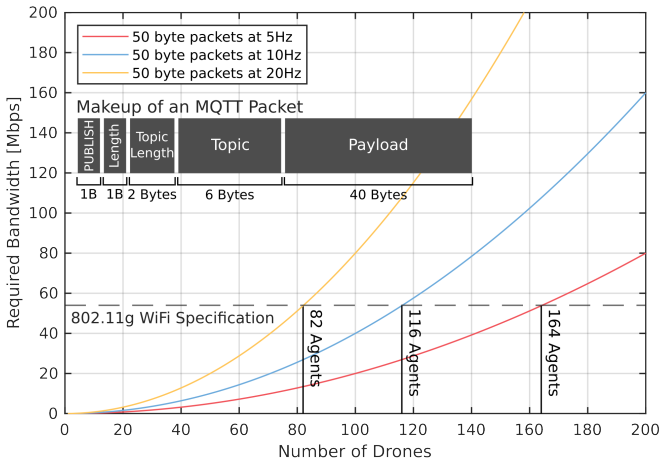

Fig. 14: Theoretical bandwidth requirements for swarms of different sizes

The geometry of a typical flight experiment for assessing the performance of a junction is shown in Figure 15. Drones are recirculated during an experiment to generate a statistically meaningful number of data points. Typically 3 drones were used on each branch and each drone circulated six times. Test geometry size was based on a compromise between compactness to improve experimental productivity and spatial resolution in terms of the number of deconflicted agents that could occupy the junction envelope at any given time. An illustration of different junction envelope radii used in the experiments and the maximum capacity in terms of cross sectional packing of deconflicted agents is shown in Figure 16.

Experimental results involving velocity are non dimensionalised via the agent cruise speed and results involving length via the agent deconfliction radius.

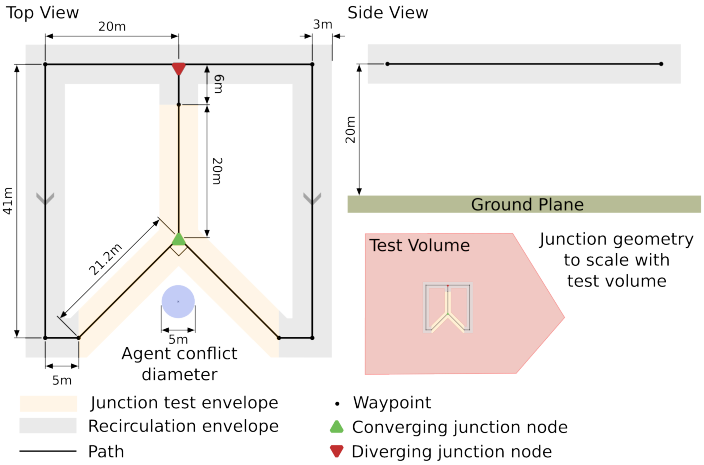

Fig. 15: Definition of test geometry for a recirculating junction experiment. Diverging node is due North of converging node.

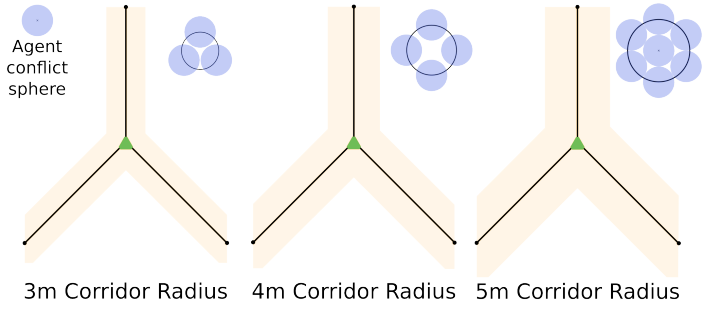

Fig. 16: Geometry of recirculating junction experiments to scale with the conflict radius of an agent. A front view of each corridor is shown with the maximum number of conflict radii which can fit in the cross-section. The corridor envelope is shown in yellow and the conflict radius is shown in blue.

For safe operations, flight testing required a crew of a minimum of two people, with one person responsible for vehicle flight management and experiment control using the GCS, and a second person responsible for manual control of individual drones using dedicated radio control transmitters should an emergency arise that can not be managed by the GCS. Flight tests were undertaken in wind speed conditions from 0 up to 25 km/h (7 m/s), which represents maximum wind speed at which drones could reliably hold position. Tests were conducted at air temperature down to 0 degrees Celsius. However flight at low temperatures reduced experimental productivity due to temperature related reduction in battery capacity. Flights were not undertaken during precipitation. A typical flight operation would involve the following steps: 1) Manually position drones in their ground start locations and power up, 2) upload experiment plans to each drone in form of JSON files, 3) Initiate drone take off and go to experiment start position 4) Initiate flight experiment, 5) Stop experiment and return drones to land at takeoff position and download log files. To simplify operations, a method was developed to safely position the aerial robots in their experiment start positions without relying on active deconfliction using only goto commands included in PX4, Figure 18. Agents are separated vertically by a pre-defined separation distance before flying to the desired x-y positions and finally moving to the desired altitude.

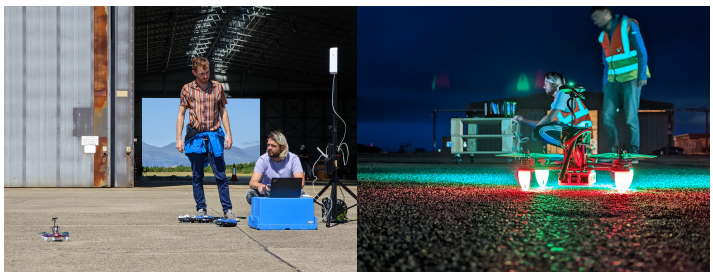

Fig. 17: Photos captured during experimental testing at Snowdonia Aerospace Centre.

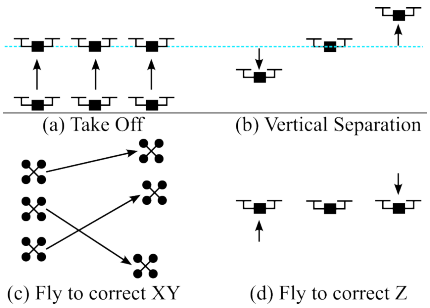

Fig. 18: Centrally controlled sequence to get drones safely into start positions without using active deconfliction (Knox et al., 2022).

4 Results

A summary of the results from converging junction experiments showing trajectories and computed congestion (mean deconfliction velocity) for different corridor radii is shown in Figure 19. The most significant region of congestion is around the node of the junction, as expected. Increasing the corridor radius increases the spatial distribution of the agent paths but otherwise trajectories are generally similar. Of note is that agents tend to travel on the outside surface of the corridor with respect to the rotation axis of the circuit. This leads to the mean paths from each circuit crossing ahead of the junction node, which is non-ideal behaviour from a deconfliction point of view. This issue is also related to the relatively simple projected distance algorithm for deciding when a waypoint has been achieved. For this particular node, changing to increment upon reaching the Easterly coordinate of the waypoint would bring forward the switching point of both left and right hand circuits and reduce the closing velocity at the point of intersection, and hence reduce the intensity of deconfliction required.

Measurements of the integral of deconfliction and merge velocities for each of the six circuits in the junction experiments referred to above are shown in Figure 20. Integration is with respect to time over one circuit and is *per agent*. Velocities are presented in dimensionless form where a value of unity is a guidance demand velocity equal to the vehicle cruise velocity. Recorded deconfliction and merge velocities are significantly higher in the first circuit due to the influence of the use of agent starting locations that were not within the circuit. In subsequent plots, *mean* integrated velocities are presented based on circuits 2-6 only.

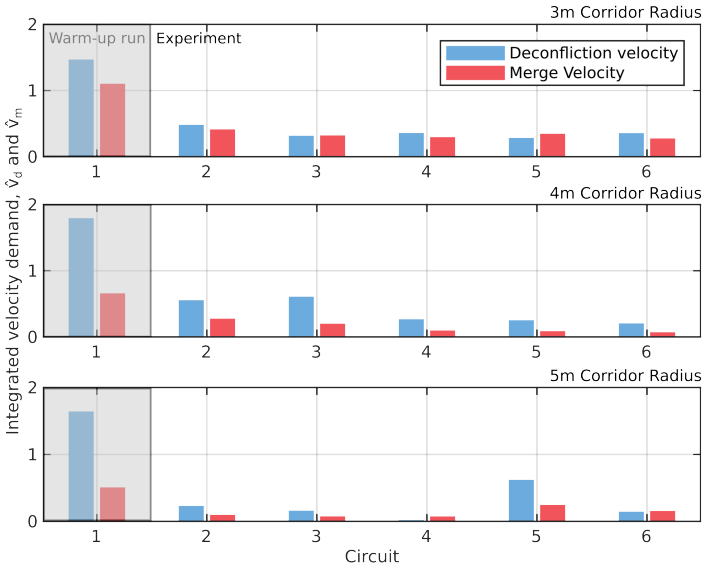

Fig. 20: Integrated non-dimensional deconfliction and merge velocities for each circuit through the 3 m, 4 m and 5 m corridor radii junctions. The first circuit has relatively higher deconfliction and merge velocities due to increased proximity at start and is not included in averaged quantities.

The effect of changing corridor radius on mean integrated deconfliction and merge velocities is shown in Figure 21. The trend is expected in that these quantities reduce with increasing corridor radius, which reduces the density of agents (reduces conflict) and relaxes the lateral manoeuvering required to stay within a corridor cross section. The trend with changing corridor radius is not linear, and in particular there is some feature of the combined geometry of 4 m radius case evident from Figure 19 that introduces additional deconfliction requirements.

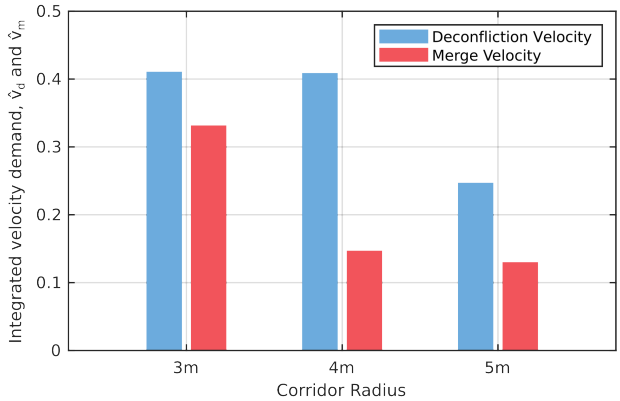

Fig. 21: Integrated mean non-dimensional deconfliction and merge velocity for 3 m, 4 m and 5 m corridor radii averaged over circuits 2-6 (steady state).

A detailed time history of proximity (distance between agents) and congestion (mean deconfliction velocity) for one circuit of a single pair of agents is shown in Figure 22. This result serves to verify the implementation of deconfliction in that mean deconfliction velocity demand is non zero when the separation distance between any two agents is less than the agent conflict diameter, as required. The deconfliction veloc-

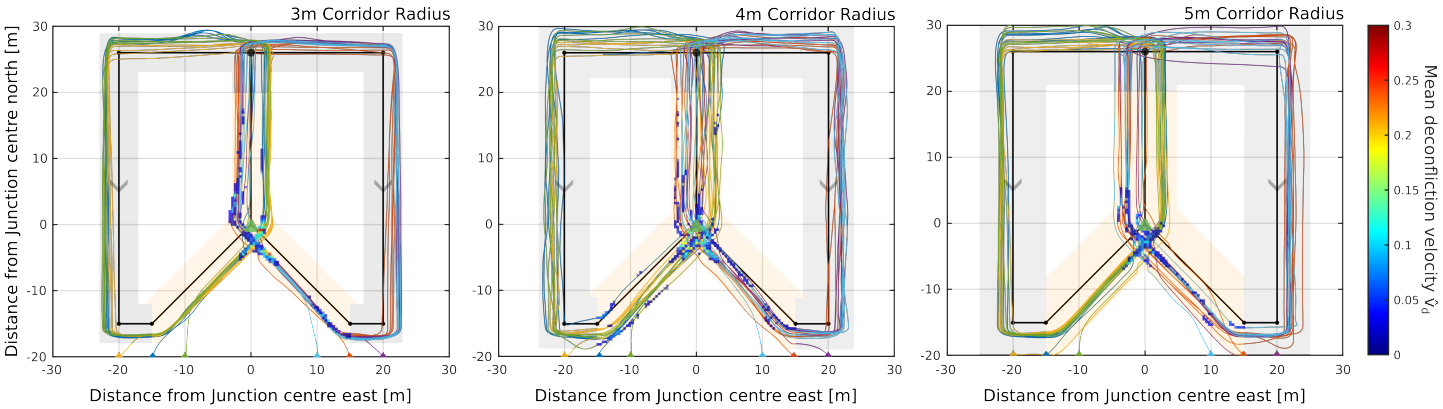

Fig. 19: Paths of each agent through the recirculating junction experiment for corridor radii of 3 m, 4 m and 5 m overlaid with deconfliction velocity. Deconfliction data only shown for deconfliction velocity greater than 0. Each loop contains three agents (total of 6 agents in experiment). Experiment is run for six complete circulations through the junction. Agent conflict radius was 5 m, shown to scale in inset. The yellow envelope represents the junction experiment test region. Grey is the return path.

ity has a large peak prior to the junction node then a smaller one afterwards. The second peak is due to influence of the change in direction once the junction node waypoint has been reached.

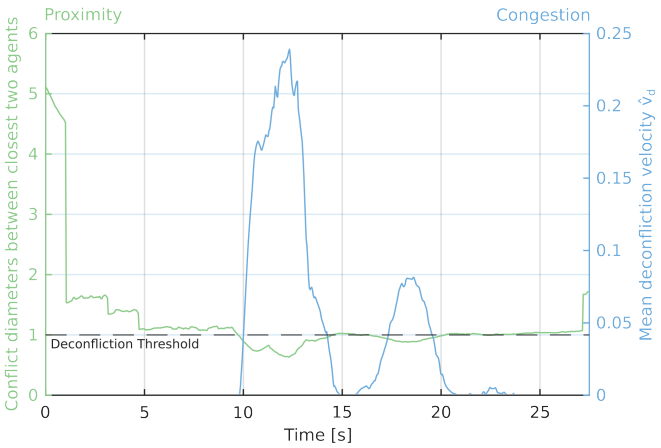

Fig. 22: Time history of the proximity between the two closest agents (green) and mean deconfliction velocity for all 6 agents (blue) for one circuit of a junction experiment with corridor radius of 3 m. The horizontal dashed line represents the proximity at which deconfliction is triggered.

The cycle-averaged deconfliction velocity per agent as a function of distance through the junction is shown in Figure 23. The trends for the different corridor radii are similar with a small peak in congestion around the waypoint at the entrance to the junction and a larger peak at the junction node. There remains evidence of the double peak around the junction node for the 3 m radius case after cycle averaging, however this feature is less evident in the 4 m and 5 m cases.

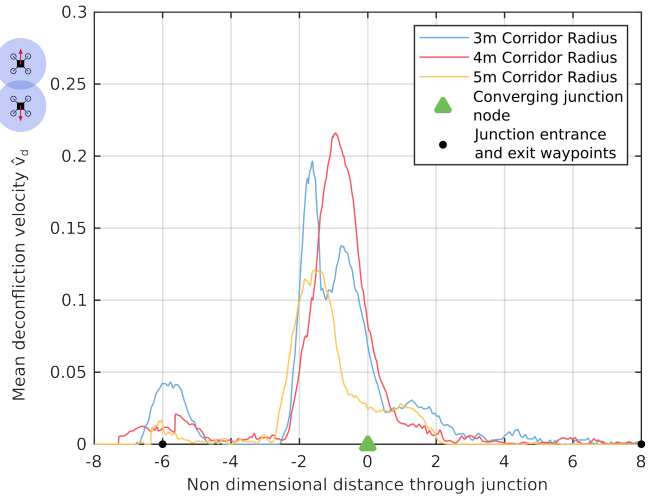

Fig. 23: Mean non-dimensional deconfliction velocity (congestion) against distance through junction for converging junction experiments for different corridor radii. Distance through junction is measured on line of symmetry normalised by conflict radius of 2.5 m.

The cycle-averaged merge velocity per agent as a function of distance through the junction is shown in Figure 24. The merge peaks coincide approximately with the entrance waypoint and the junction node, but with the peak being slightly ahead at the entrance and behind at the node. The peak merge velocity is highest for the smallest corridor radius, consistent with the need to follow a tighter turn radius to stay within the corridor. A direct comparison of deconfliction velocity and merge velocity for the 3 m corridor radius case plotted on the same axes is shown in Figure 25. The fact that the deconfliction peak is substantially ahead of the merge peak at the junction node confirms that it is the crossing streams effect ahead of the node that is causing the high deconfliction rate rather than the funnelling effect of the converged corridors.

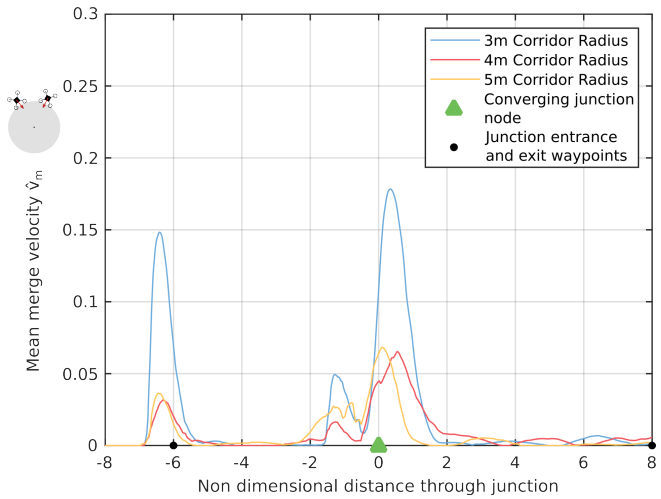

Fig. 24: Mean non-dimensional merge velocity against distance through junction for different corridor radii.

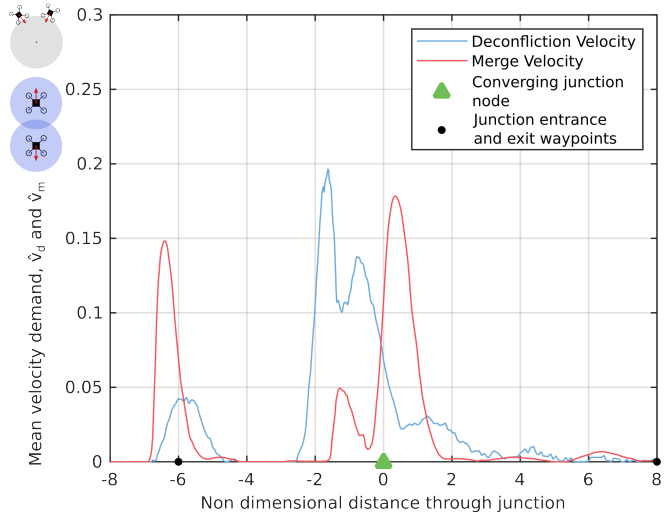

Fig. 25: Comparison of mean deconfliction and merge velocities against distance through junction for a corridor radius of 3 m.

We now consider results from a more sophisticated converging junction concept in which the agents travel in a helical fashion on the surface of cylindrical corridors. The corridors are joined at a junction node using the principle of meshing helical gears such that agents are able to transfer (switch) from one cylinder to an adjacent meshing cylinder without having to cross paths with any other agents. Agent trajectories from a MITL simulation of the junction is shown in Figure 26. Experimental measurements of mean deconfliction velocity through the junction are shown in Figure 27, with comparison to an equivalent simple converging junction. The magnitude of separation velocity for the meshed helical junction is significantly less than that for a simple converging junction, as expected. However, for the helical junction, improved deconfliction does come at the cost of decreased transport efficiency as the helicity increases the effective path length and hence transit time through the junction.

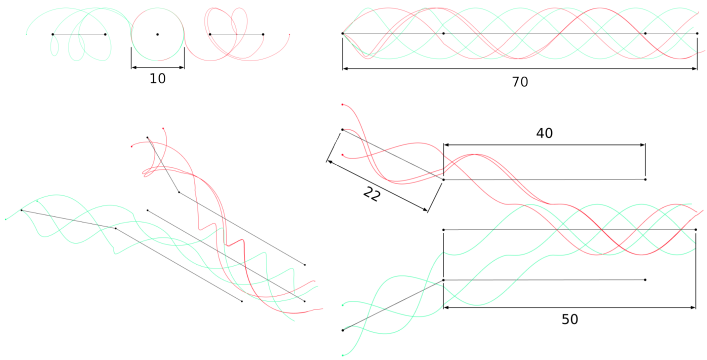

Fig. 26: Definition of the helical junction geometry used in Figure 27. Dimensions in m.

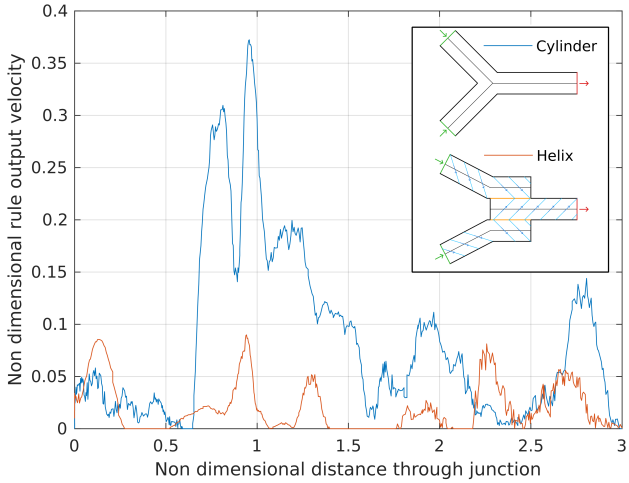

Fig. 27: Comparison of deconfliction velocity for a simple converging junction and a meshed helical junction.

The correct ensemble behaviour of the agents was verified by flying a number of different paths in simulation and in the real world. Figure 28 shows the MITL simulation output side by side with a long exposure photograph of the flight test. There are some small differences due to real-world effects such as communication latency and sensor measurement error however the ensemble behaviour is clearly correct.

5 Conclusion

Design of field trials for testing deconfliction of aerial robots with line of sight constrained test volumes is driven by satisfying conflicting requirements for maximising spatial resolution (maximising the number of deconflicted agents that can be packed in the test volume by minimising the conflict radius) and maximising experiment time efficiency by maximising the vehicle cruise speed. Minimum deconfliction radius is set by the distance given by the product of latency and cruise speed and the stopping distance at maximum deceleration, hence spatial resolution for a given cruise speed and vehicle acceleration performance is increased by minimising latency. For the experimental setup used, there was a baseline delay of 0.05 seconds due to centralised message brokering that was approximately independent of the number of drones (up to

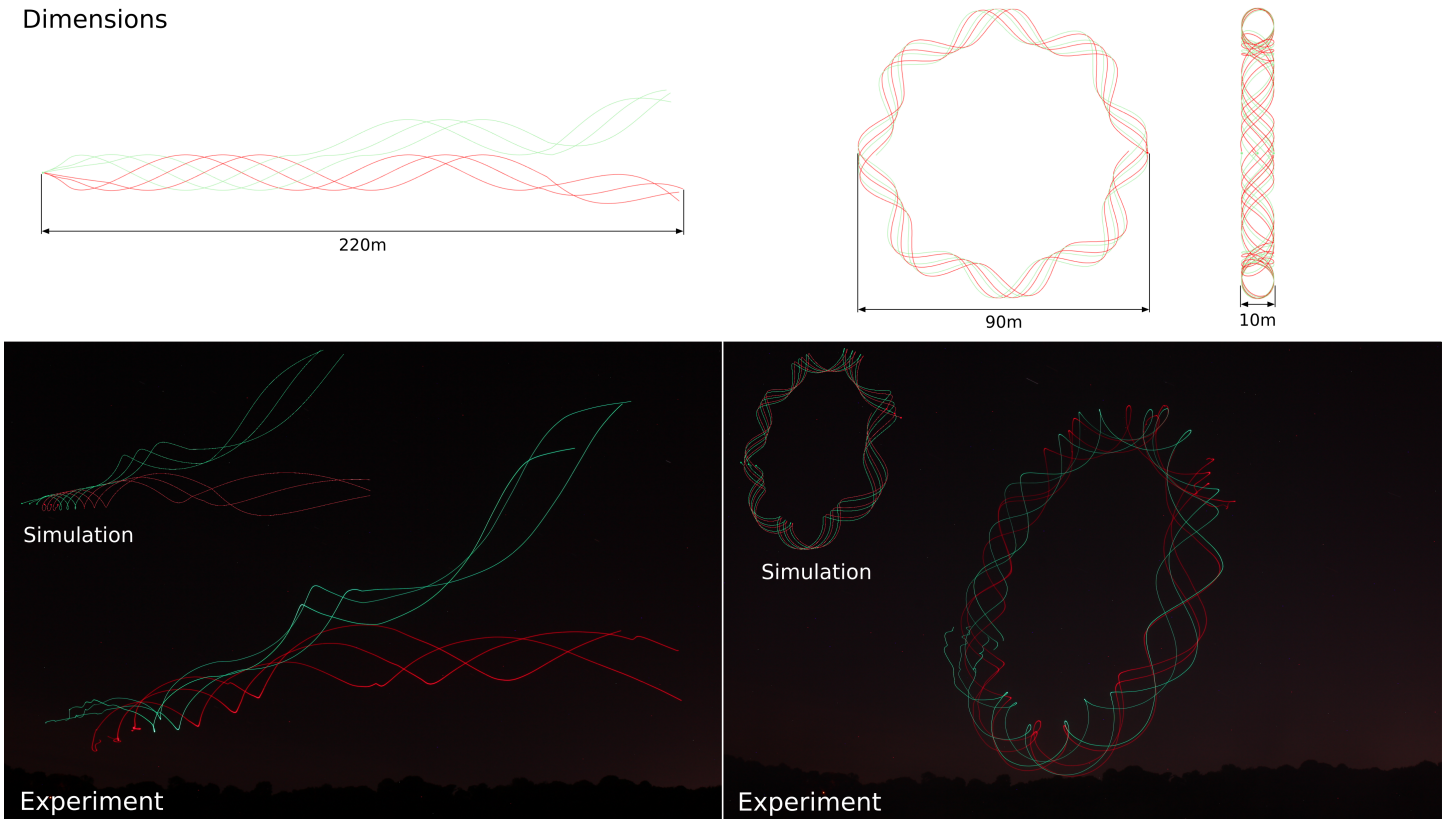

Fig. 28: Long exposure photo of a flight test next to the output from MITL simulation to verify correct ensemble behaviour.

10 drones) and a WiFi delay that increased roughly proportional to the number of drones. The mean latency with 6 drones was approximately 0.1 seconds with a standard deviation of 0.05 seconds. Further delays are introduced due to asynchronous update of guidance and communication frames at 10 Hz giving a total worst case latency of around 0.4 seconds. A choice was made to operate experiments at cruise speed of 2 m/s, which required a deconfliction radius of 2.5 m to satisfy worst case stopping criteria. A higher cruise speed for the same conflict radius would have allowed greater experimental productivity in terms of the amount of useful data that could be obtained between battery changes. With the present constraints, increasing the cruise speed also increases the required conflict radius and hence has neutral effect on experimental productivity. Mean latency could be reduced in future by adopting a mesh networking approach that reduces the centralised communication burden, however this may not address the peak latency, which is what drives the required deconfliction radius. The total number of drones flown simultaneously in the present experiment was a relatively modest six. From an available WiFi bandwidth perspective, it should be possible in theory to fly up to 116 drones with 10 Hz communication frame rate in the same experimental setup however larger conflict radii or lower cruise speeds would need to be used due to increased latency.

An experiment was successfully conducted to evaluate deconfliction within a converging junction between two traffic corridors with three vehicles continuously circulating on each of two different loops to improve data productivity, with experiments repeated for traffic corridors of different radius. The principal experimental measurements for each drone were position, deconfliction velocity demand and (corridor) merge velocity demand. Aggregate junction performance in terms of congestion was measured based on the integral of deconfliction velocity per drone per cycle through the junction. Integrated merge velocity per drone per cycle provided evidence of the equivalent impact of corridor following on guidance demand. Increasing corridor radius decreased congestion and aggregate merge velocity as expected. Due to the relatively low density of drones in each circuit and that turns were all of the same handedness, drones paths tended to be around the outside edge of the corridor for each circuit, resulting in path crossing just ahead of the junction node. This generated a region where drones had high closing velocity in close proximity and hence the deconfliction activity was high. The experiment could be improved in this respect by adapting the merge rule logic to ensure a more uniform distribution across the corridor or by introducing corridor curvature of opposite sign to align the principal flow with the centreline. Evaluation of a more sophisticated junction concept using meshed helical corridors with implicit velocity alignment significantly reduces congestion compared to non velocity aligned junctions, albeit with agents taking more time to transition through the junction. A complete torroidal helical corridor was implemented experimentally as an example of building block for development of composite helical interchanges.

Acknowledgment

This work was part of the EPSRC CASCADE Program EP/R009953/1.

Long exposure photographs taken by Dan Koning.

References

Airbus. (2018). *Blueprint for the Sky* (tech. rep.). Airbus.

Bauranov, A., & Rakas, J. (2021). Designing airspace for urban air mobility: A review of concepts and approaches. *Progress in Aerospace Sciences*, *125*, 100726.

Berna-Koes, M., Nourbakhsh, I., & Sycara, K. (2004). Communication efficiency in multi-agent systems. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, *2004*(3), 2129–2134.

Besada, J. A., Carramiñana, D., Bergesio, L., Campaña, I., & Bernardos, A. M. (2022). Modelling and Simulation of Collaborative Surveillance for Unmanned Traffic Management. *Sensors*, *22*(4), 1498.

Capitán, C., Pérez-León, H., Capitán, J., Castaño, Á., & Ollero, A. (2021). Unmanned Aerial Traffic Management System Architecture for U-Space In-Flight Services. *Applied Sciences*, *11*(9), 3995.

Carramiñana, D., Campaña, I., Bergesio, L., Bernardos, A. M., & Besada, J. A. (2021). Sensors and Communication Simulation for Unmanned Traffic Management. *Sensors*, *21*(3), 927.

Federal Aviation Administration. (2020). *Unmanned Aircraft System (UAS) Traffic Management (UTM) Concept of Operations, V2.0* (tech. rep.).

Federal Aviation Administration & NASA. (2020). *Urban Air Mobility (UAM) Concept of Operations v1.0* (tech. rep.). FAA.

Grabe, V., Riedel, M., Bulthoff, H. H., Giordano, P. R., & Franchi, A. (2013). The TeleKyb framework for a modular and extendible ROS-based quadrotor control. *2013 European Conference on Mobile Robots*, 19–25.

Hattenberger, G., Bronz, M., & Gorraz, M. (2014). Using the paparazzi UAV system for scientific research. *IMAV 2014, International Micro Air Vehicle Conference and Competition 2014*, pp–247.

Hoekstra, J., Kern, S., Schneider, O., Knabe, F., & Lamiscarre, B. (2015). Metropolis – Concept design. *341508*, 1–56.

Jang, D. S., Ippolito, C., Sankararaman, S., & Stepanyan, V. (2017). Concepts of airspace structures and system analysis for UAS traffic flows for urban areas. *AIAA Information Systems-AIAA Infotech at Aerospace, 2017*, (January), 1–15.

Knox, S. J. C., Crann, V. J., Amiri, P., & Crowther, W. J. (2022). A practical framework for multi agent experiments in aerial robotics. *2022 7th International Conference on Mechanical Engineering and Robotics Research (ICMERR)*, 108–113.

Mellado-Bataller, I., Pestana, J., Olivares-Mendez, M. A., Campoy, P., & Mejias, L. (2013). MAVwork: A Framework for Unified Interfacing between Micro Aerial Vehicles and Visual Controllers. In *Studies in computational intelligence* (pp. 165–179).

Millan-Romera, J. A., Acevedo, J. J., Castano, A. R., Perez-Leon, H., Capitan, C., & Ollero, A. (2019). A UTM simulator based on ROS and Gazebo. *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, 132–141.

Molina, M., Carrera, A., Camporredondo, A., Bavle, H., Rodriguez-Ramos, A., & Campoy, P. (2020). Building the executive system of autonomous aerial robots using the Aerostack open-source framework. *International Journal of Advanced Robotic Systems*, *17*(3), 172988142092500.

Nguyen, C. P., & Flueck, A. J. (2011). Modeling of communication latency in smart grid. *2011 IEEE Power and Energy Society General Meeting*, 1–7.

Nguyen, D. D., Rohacs, J., & Rohacs, D. (2021). Autonomous Flight Trajectory Control System for Drones in Smart City Traffic Management. *ISPRS International Journal of Geo-Information*, *10*(5), 338.

Pasandideh, F., João, —., Da Costa, P. J., Kunst, —. R., Hardjawana, W., & Pignaton De Freitas, —. E. (2023). A systematic literature review of flying ad hoc networks: State-of-the-art, challenges, and perspectives. *Journal of Field Robotics*.

Preiss, J. A., Honig, W., Sukhatme, G. S., & Ayanian, N. (2017). Crazyswarm: A large nano-quadcopter swarm. *Proceedings - IEEE International Conference on Robotics and Automation*, 3299–3304.

Quan, Q., & Li, M. (2020). Sky highway design for dense traffic. *ArXiv*, *abs/2010.09159*.

Ramee, C., & Mavris, D. N. (2021). Development of a Framework to Compare Low-Altitude Unmanned Air Traffic Management Systems. *AIAA Scitech 2021 Forum*, *1 PartF*, 1–24.

Remes, B., Hensen, D., van Tienen, F., De Wagter, C., van der Horst, E., & de Croon, G. (2013). Paparazzi: how to make a swarm of Parrot AR Drones fly autonomously based on GPS. *Imav 2013*, (September), 17–20.

Sanchez-Lopez, J. L., Molina, M., Bavle, H., Sampedro, C., Suárez Fernández, R. A., & Campoy, P. (2017). A Multi-Layered Component-Based Approach for the Development of Aerial Robotic Systems: The Aerostack Framework. *Journal of Intelligent & Robotic Systems*, *88*(2-4), 683–709.

Sanchez-Lopez, J. L., Suarez Fernandez, R. A., Bavle, H., Sampedro, C., Molina, M., Pestana, J., & Campoy, P. (2016). AEROSTACK: An architecture and open-source software framework for aerial robotics. *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 332–341.

Silano, G., & Iannelli, L. (2020). CrazyS: A Software-in-the-Loop Simulation Platform for the Crazyflie 2.0 Nano-Quadcopter. In *Studies in computational intelligence* (pp. 81–115). IEEE.

Sunil, E., Hoekstra, J., Ellerbroek, J., Bussink, F., Nieuwenhuisen, D., Vidosavljevic, A., & Kern, S. (2015). Metropolis: Relating airspace structure and capacity for extreme traffic densities. *Proceedings of the 11th USA/Europe Air Traffic Management Research and Development Seminar, ATM 2015*, (June).

Tony, L. A., Ratnoo, A., & Ghose, D. (2020). Corridrone: Corridors for drones, an adaptive on-demand multi-lane design and testbed. *CoRR*, *abs/2012.01019*.

Van De Loosdrecht, J., Dijkstra, K., Postma, J. H., Keuning, W., & Bruin, D. (2014). Twirre: Architecture for

autonomous mini-UAVs using interchangeable commodity components. *International Micro Air Vehicle Conference and Competition*, (August).

Zhao, Z., Luo, C., Zhao, J., Qiu, Q., Gursoy, M. C., Caicedo, C., & Basti, F. (2019). A Simulation Framework For Fast Design Space Exploration Of Unmanned Air System Traffic Management Policies. *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, 1–10.