Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Review

An ontology-based system to avoid UAS flight conflicts and collisions in dense traffic scenarios

David Martín-Lammerding[a], José Javier Astrain[a,b,*], Alberto Córdoba[a], Jesús Villadangos[a,b]

[a] Department of Statistics, Computer Science and Mathematics, Universidad Pública de Navarra, Campus de Arrosadia, 31006 Pamplona, Spain
[b] Institute of Smart Cities, Universidad Pública de Navarra, Campus de Arrosadia, 31006 Pamplona, Spain

ARTICLE INFO

Keywords:
UAS
Ontology
Autonomous
Collision avoidance systems
Knowledge
Situational-awareness

ABSTRACT

New Unmanned Aerial Systems (UAS) applications will increase air traffic densities in metropolitan regions. Collision avoidance systems (CAS) are a key component in integrating a high number of UAS into the airspace in a safe way. This paper presents a distributed, autonomous, and knowledge-based CAS, called Dronetology System (DroS), for UASs. The CAS proposed here is managed using a novel ontology, called Dronetology-cas, which allows to make autonomous decisions according to the knowledge inferred from the data gathered by the UAS.

DroS is deployed as part of the payload of the UAS. So, it is designed to run in an embedded platform with limited processing capacity and low battery consumption. DroS collects data from sensors and collaborative elements to make smart decisions using knowledge obtained from collaborative UASs, adapting the maneuvers of the aerial vehicles to their original flight plans, their kind of vehicle, and the collision scenario. DroS accountability involves recording its internal operation to assist with reconstructing the circumstances surrounding an autonomous maneuver or the details previous to a collision. DroS has been verified using the hardware in the loop (HIL) technique with a UAS traffic environment simulator. Results obtained show a significant improvement in terms of safety by avoiding collisions.

1. Introduction

The use of Unmanned Aerial Systems (UAS) improves efficiency in many industrial areas like logistics and infrastructure inspection. It also avoids the risk of manned missions in emergency situations and hazardous missions. However, there are multiple requirements to conduct a flight. Aeronautical authorities have to review the UAS operation and publish a NOTAM (Falato, 1988) for authorized flights. In order to enable the development of commercial services, it is necessary to expand the authorized flight zones, avoid segregating airspace and increase the autonomy of UAS.

UAS Traffic Management (UTM) and the automation of mission authorization processes are being addressed in different projects, as described in Barrado et al. (2020) and Lieb and Volkert (2020). Both of them propose management flexibilization techniques to minimize risks when sharing airspace. However, some emergency situations, such as flight delays due to unforeseen events, may cause traffic management not to be sufficient enough to avoid collisions.

Aeronautical authorities like EASA in Europe are worried about flight safety. Thus, flight authorization processes require a compulsory risk assessment reports following the Specific Operations Risk Assessment (SORA) methodology (JARUS, 2019) to evaluate the risk of the operation. This methodology provides a risk assessment tool and lists a set of mitigation actions to reduce the operation's risk. One of the main mitigation actions proposed in SORA is the deployment of Collision Avoidance Systems (CAS) for each UAS. In addition to integrating UAS into a common airspace, another challenge avoiding collisions between UAS in dense traffic airspace. The fundamental see-and-avoid capability of manned aircraft has to be adapted to detect and avoid other aircraft in the UAS environment. From an unmanned aircraft systems (UTM) traffic management point of view, intelligent collision-free trajectory planning is required. Centralized UTMs have a complete view of the system, but decisions could be delayed, increasing the risk of collision. Collision avoidance is a local decision for each UAS, but collaborating with other UAS in the same area. Then, distributed decision systems deployed on UASs can make decisions locally, solving this problem, although local processing could add its own delay.

* Corresponding author at: Department of Statistics, Computer Science and Mathematics, Universidad Pública de Navarra, Campus de Arrosadia, 31006 Pamplona, Spain.
E-mail addresses: david.martin@unavarra.es (D. Martín-Lammerding), josej.astrain@unavarra.es (J.J. Astrain), alberto.cordoba@unavarra.es (A. Córdoba), jesusv@unavarra.es (J. Villadangos).

https://doi.org/10.1016/j.eswa.2022.119027
Received 12 June 2022; Received in revised form 10 October 2022; Accepted 10 October 2022
Available online 18 October 2022

Security and safety are top priorities for the aeronautical sector. Commercial aviation safety is increasing over the years thanks to the accountability that involves maintaining an accurate registration of flight data, maneuvers, and conflicts. The same accountability should apply to UASs to improve safety in their missions. As occurs with commercial aviation, UAS's flights should be tracked and recorded.

The black box is a piece of essential airborne equipment for modern commercial aviation aircraft that provide flight recording capabilities and a cockpit voice recorder. UAS's black boxes should record both flight data (FDR), and messages exchanged. Some black box intended for UAS are Holdings (2022), Tl-Elektronic (2022) and UAV Navigation (2022). However, they present limitations like limited interfaces and the inability to record the decision-making processes of an autonomous UAS.

The design of an autonomous system should consider accountability requirements for information systems as presented in Feigenbaum, Jaggard, and Wright (2011) and detailed in Julisch, Suter, Woitalla, and Zimmermann (2011). Semantic-based systems allow inference debugging, accountability and explainability as described in Barredo-Arrieta et al. (2020), Chari, Gruen, Seneviratne, and McGuinness (2020) and Chari, Seneviratne, et al. (2020).

The collision avoidance problem for a scenario with multiple UAS follows three strategies to address the conflict resolution problem, as described in Sánchez, Casado, and Bermúdez (2020): before the mission starts, collaborating during the flight or/and in an independent strategy. The *pre-plan* strategy focuses on pre-flight planning to avoid collisions. A central system collects all the available information on UAS and organizes flights to avoid collisions. This strategy has two major drawbacks: it does not scale as the number of UAS grows, and it does not adapt to unexpected contingencies. A *collaborative* strategy is applied between several UAS. It consists of sharing their status and plans to accord a maneuver that is intended to solve the conflict. In order to implement a collaborating strategy, a transponder is required in each aircraft. Position and bearing values are received from and broadcasted to any other UAS equipped with a transponder within its coverage radius. ADS-B is one of the standards for collaborative systems based on sharing location information obtained from the GPS system, as described in Hein (2000).

FLARM (FLARM, 2022) is another widely used collaborative system in Europe. In an *autonomous* strategy, each aircraft makes decisions based on data gathered from its own sensors on board, like video cameras detecting obstacles without requiring interaction with any external system. This defines a non-collaborative UAS. Multiple technologies have been applied to non-collaborative systems such as vision cameras in Zuehlke, Prabhakar, Clark, Henderson, and Prazenica (2019), LIDAR in Papa, Ariante, and Del Core (2018), sonar in Papa (2018), radar in Gellerman, Mullins, Foerster, and Kaabouch (2018), etc. A survey on obstacle detection technologies is presented in Muraru (2011).

In this paper, we present a novel distributed and autonomous, knowledge-based CAS for UAS, called DroS. Each UAS collects information from its local sensors and gathers collaborative information from other UASs with its transponder. DroS transforms this information into knowledge using the ontology Dronetology-cas to take decisions locally. DroS' knowledge increases as the UAS flies, so decisions are improved and collisions are avoided or minimized. UASs require real-time responses for real use, so the inference process is performed at the own UAS.

Furthermore, collaborative and autonomous collision avoidance strategies based on Dronetology-cas are proposed. DroS is implemented and deployed in a HIL simulation to verify the developed avoidance strategies in dense traffic scenarios.

The rest of the paper is structured as follows. Section 2 presents the related work and the contributions of the paper. Section 3 describes DroS Foundations. Section 4 describes DroS inference for collision avoidance, and Section 5 describes DroS implementation. Section 6 analyzes DroS conflict avoidance. Section 7 summarizes the simulation results obtained. Section 8 presents the conclusions and future work. Finally, Acknowledgment, Appendixes, and References end the paper.

2. Related work

CASs for UAS have been widely studied with different approaches. Early works apply CAS's techniques for manned aircraft to UAS. Another approach is the application of static obstacle avoidance techniques for mobile robots, adapting them to a 3D scenario. In an alternative way, we follow a general taxonomy to review the most popular techniques, based on four fundamental tasks performed by CAS for UAS: surveillance, coordination, maneuver, and autonomy, as proposed in Jenie, Van Kampen, Ellerbroek, and Hoekstra (2016).

Surveillance, in a CAS context, refers to the detection of conflicts and the prediction of collisions. It can be performed in three main ways: centralized, distributed, or multiple sources. Table 1 shows sensing environment types and conflicts detection methods.

A centralized surveillance system collects and distributes data from a central-ground station such as an Air Traffic Control (ATC) or an Aviation Weather Center (AWC). On the other hand, distributed surveillance can be collaborative or non-collaborative. A collaborative surveillance system collects data from every UAS that broadcasts its flight data. A non-collaborative surveillance system is performed with on board sensors, independently of other UAS. Multiple surveillance sources (surveillance combination) is a combination of two or more of the previous surveillance systems.

Data fusion techniques allow merging and leveraging of data collected. Data fusion techniques build a unified image of the environment from multiple sensors and/or collaborative elements (Hall & Llinas, 1997). Data fusion applied to CAS for UAS has been addressed in Fasano et al. (2015b) and Graham et al. (2011). Fasano et al. (2015b) presented a non-cooperative anti-collision system using data fusion of a Ka-band pulsed radar and optical sensors. Graham et al. (2011) presented another data fusion-based solution, in this case, merging radar and ADS-B data. In both cases, data fusion improves conflict detection and location accuracy. However, the fused data is not used to infer new knowledge about the characteristics of remote aircraft and their expected behavior.

A multi-sensor data fusion implementation is presented in Jackson, Bošković, and Diel (2015) to avoid the use of ADS-B and overcome its limitations described in Langejan, Sunil, Ellerbroek, and Hoekstra (2016) and McCallie, Butts, and Mills (2011).

Data fusion from different sources is presented in Ramasamy, Sabatini, and Gardi (2014), combining non-collaborative and collaborative technologies applied to small UASs. The study performs simulations of their operation with up to three UAS simultaneously. However, more UAS traffic is expected, so scenarios with more UAS in conflict should be considered. All the above proposals integrate sensors' collected information in real-time. Knowledge acquisition from information gathered by sensors is not considered. However, knowledge is a great advantage in making decisions at any step of the CAS process.

Collision prediction is the other part of the surveillance function performed by a CAS. Table 2 summarizes the most common techniques reviewed in Chand, Mahalakshmi, and Naidu (2017). Constant Bearing Decreasing Range (CBDR) stands out for its low computational cost.

Other collision detection techniques are based on Particle Filters (Bugallo, Xu, & Djurić, 2007) and Kalman Filter in Peach (1995) and Shakernia, Chen, and Raska (2005). Both require multiple samples which implies either a high sampling frequency or waiting to have enough data. This implies that both approaches would increase the computer processing requirements or reduce the time available to react.

Coordination avoidance is a CAS task so that UASs agree on the maneuvers for each UAS that would simplify the resolution of the conflict. Three types of coordination are considered, explicit, implicit, and non-existent. Explicit coordination requires communication among the UASs in conflict. It is detailed in Cunningham, Wu, Biaz, and Jones (2013) and Sislak, Rehak, Pechoucek, Pavlicek, and Uller (2006). Both are decentralized systems. Conflict resolution in Sislak et al. (2006) is

Table 1
Types of environment sensing and conflict detection.

Surveillance	Remarks	Examples	Ref
Centralized	Available before the start of the flight	ATC, AWC	Brooker (2013), Nikolos, Valavanis, Tsourveloudis, and Kostaras (2003)
Collaborative	Suitable for large and medium UAS	TCAS, ADS-B, FLARM	Lin and Saripalli (2015), Strobel and Schwarzbach (2014)
Non-collaborative active	Suitable for all types of UAS	Radar, Laser	Kendall et al. (2017)
Non-collaborative passive	Widely used in small UAS	Electro Optical (EO) Acoustic	Gao et al. (2020), Kendall et al. (2017)
Surveillance combination	Situational awareness improvement	ADS-B + EO	Fasano, Accardo, Tirri, Moccia, and De Lellis (2015b), Graham et al. (2011)

Table 2
Collision detection techniques.

Techniques	Remarks	Computational cost	Ref
Markov Decision Process (MDP)	Based on discretizing the state space into some grid points	High	Fu, Yu, and Zhang (2015), Ong and Kochenderfer (2017)
Worst Case Approximation (WCA)	Evaluate all the possible intruder trajectories	High	Strobel and Schwarzbach (2014)
Closest Point of Approach (CPA)	Paths are estimated based on the current trajectories and their respective positions and speeds	Medium	Zarandy, Zsedrovits, Pencz, Nameth, and Vanek (2015)
Constant bearing, decreasing range (CBDR)	A special case of CPA in which the distance decrease and the relative bearing is constant	Low	Fasano, Accardo, Tirri, and Moccia (2016), Fasano, Accardo, Tirri, Moccia, and De Lellis (2015a)

based on sharing the flight plan and implementing a protocol to negotiate the resolution of the conflict. Connectivity between the UASs and conflict negotiation requires time which is a problem in situations that require a short reaction time. Another decentralized system is described in Cunningham et al. (2013), which negotiates conflict resolution. The implementation is based on the XBee protocol. No simulations with dense UAS traffic were documented. Explicit decentralized coordination is performed in Vera, Cobano, Alejo, Heredia, and Ollero (2015). They simulate up to nine conflicting UAS, but the response time of the algorithm limits its implementation. ACAS Xu (Owen, Panken, Moss, Alvarez, & Leeper, 2019) implements a CAS with explicit coordination for UAS, derived from the Traffic Alert & Collision Avoidance System (TCAS) (Williamson & Spencer, 1989).

Implicit coordination implies that UASs in conflict share a common set of rules to define a maneuver. This type of coordination requires information exchanges that may influence subsequent maneuvers of other UAS. Implicit coordination presented in Capitán, Merino, and Ollero (2011) improves UAS performance. However, inter-communications issues are not studied. Uncoordinated avoidance implies that each UAS involved in the task has its independent maneuver. This chaotic situation could lead to unexpected situations. This makes conflict resolution difficult to compute, as it has to consider every possible move of the other UAS.

After detecting a risk of collision, the CAS must calculate a maneuver to reduce conflicts and avoid collisions. Anti-collision maneuvers are classified as strategic, tactical, or evasive. The main methods to evaluate the maneuver are summarized in Table 3. This evaluation impacts the flight plan, as it will be changed. Ideally, the evaluation should minimize its effects on the original flight plan because trajectory planning for UASs has multiple constraints such as battery lifetime, environment information, and uncertainty in the flight zones.

A strategic maneuver changes the flight plan significantly to avoid dense air-traffic areas. Trajectory planning is included as a strategic maneuver. Our design does not consider strategic maneuvers because they are intended for pre-flight planning. Our proposal can avoid conflicts during the flight in a dynamic way. A tactical maneuver modifies a small part of the flight plan, while an escape or an evasive maneuver consists of abrupt direction changes. Table 3 relates the type of maneuver with a collision avoidance path calculation. Potential Field (PF) (Zhao et al., 2017) is a collision avoidance path calculation method that simulates a force field where the UAS is attracted to its final destination and repulsed from conflicts. PF avoids collisions without requiring every other UAS to implement it. That is, PF can be applied independently.

Optimal Reciprocal Collision Avoidance (ORCA) (Van Den Berg, Guy, Lin, & Manocha, 2011) is a velocity-based obstacle avoidance method framework. An evolution for real-time UAS applications is described in Alejo, Cobano, Heredia, and Ollero (2014). However, ORCA and the derived method for UAS require that every UAS in conflict implements the same method to avoid a collision. This is a limitation when UASs with different payloads share the same airspace. Isufaj et al. (2022) proposed an avoidance method based on changing UAV heading with a reinforcement algorithm that is simulated with up to four concurrent UAVs. Our proposal combines heading and speed changes, based upon knowledge available and inferred, to improve avoidance in dense traffic scenarios, reaching up to nine UAVs.

A proposal that does not modify the UAS's planned path is a commercial CAS based on PX4 and ADS-B receivers (PX4, 2022). Instead, it performs a predefined evasive maneuver, such as returning to its starting location (home) or landing. However, this CAS reduces the UAS's performance as the mission is aborted, and does not avoid conflicts when the UAS is landing or returning home.

The operation of a CAS can have two levels of automation, pilot assistance or full-autonomous. A CAS with pilot-assistance interacts with the pilot and recommends him a maneuver, so it requires that the pilot executes the maneuver recommendation. A full-autonomous CAS executes the calculated maneuver without a pilot interaction.

Pilot-executed collision avoidance is still dominant, but the increasing Beyond Visual Line of Sight (BVLOS) flights would boost the development of autonomous CAS. An example of an autonomous CAS for BVLOS flights vision-based is Iris Automation (2022). ACAS-Xu and Daidalus are two implementations of CAS for UAS that interact with a UAS pilot, as described in Davies and Wu (2018). Both of them display alerts and suggest maneuvers to the UAS pilot. However, this way of

Table 3
Collision avoidance path calculation techniques.

Techniques	Remarks	Types of maneuver	Computational cost	Ref
Mixed Integer Linear Programming	Path planning problem represented by linear constraints on a mixture of continuous and integer variable	Tactical	High	Israelsen et al. (2014), Richards and How (2002)
Dynamic Programming	Divide the optimization problem into smaller sub-problems	Tactical	High	Jorris and Cobb (2008), Sunberg, Kochenderfer, and Pavone (2016)
Potential Fields (PF)	Movement based on repel and attraction forces	Evasive	Low	Chuang and Ahuja (1998), Zhao, Jiao, Zhou, and Zhang (2017)
Search Algorithms	Search a discrete path using a cost function and constraints	TacticalEvasive	High	Karelahti, Virtanen, and Öström (2008), Kim, Gu, and Postlethwaite (2008)
Particle Swarm Optimization (PSO)	Search candidate solutions mimicking animals‘ social behavior	Evasive	Medium	Fu, Ding, and Zhou (2011), Karimi and Pourtakdoust (2013)
Markov Decision Process (MDP)	Multiagent Reinforcement Learning formalized by a MDP	TacticalEvasive	Medium	Isufaj, Omeri, and Piera (2022)

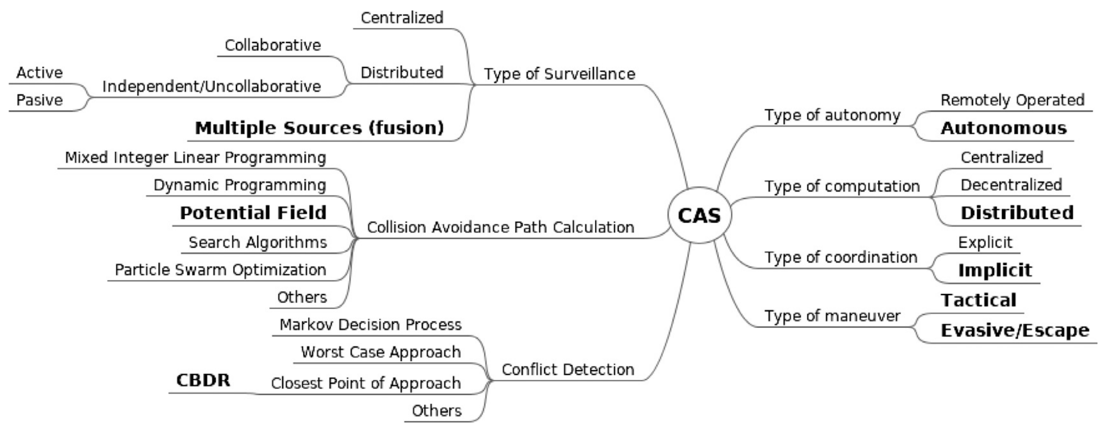


Fig. 1. CAS taxonomy. DroS design features are highlighted in bold text.

operating both is unsuitable for being implemented as an autonomous system.

Fig. 1 shows a taxonomy of CAS based on the works of Chand et al. (2017) and Jenie et al. (2016). The main contribution of this work is a distributed, autonomous, and knowledge-based CAS for UAVs that minimizes UAV collisions, called DroS. The ontology is used for knowledge inference from data retrieved from the UAV. The design, implementation, and verification of DroS are detailed in our proposal. Its semantic architecture, based on the ontology Dronetology-cas, provides autonomous decision-making based on knowledge and re-utilization of data, rules, and services. Furthermore, multiple surveillance sources integration and the implemented inference process increase the knowledge available for decision-making. This improves DroS situational awareness and its conflict avoidance.

Therefore, DroS is designed for collision avoidance between multiple UASs of any type, independently of their weight, size, or payload, as it can adapt its avoidance method to whatever situation. As a result, droS improves UAVs' safety even in dense traffic scenarios. It allows DroS-equipped UAVs to coordinate implicitly between them for avoidance maneuvers and to evade non-equipped DroS UAVs or DroS-equipped UAVs when inter-communications or time does not allow them to react. Furthermore, DroS accountability facilitates the debugging of flight encounters and also offers evidence of its operation.

DroS implementation and autonomy are verified using the simulator with HIL capabilities presented in Martín-Lammerding, Astrain, and Córdoba (2022a), showing that DroS can be implemented in an onboard PC with limited computing, and it may keep a reactive response time for avoidance purposes.

The maneuvers available in DroS are *tactical*, when implicit coordination is possible, and *evasive* for the rest of situations. DroS's anti-collision maneuvers are restricted from using altitude changes to accommodate the scenario of low-altitude UAV traffic with limited flight altitudes. However, it could include vertical maneuvers when required in other scenarios.

DroS design characteristics are highlighted in bold text in Fig. 1. The type of surveillance selected for DroS is *multiple sources* and the *CBDR* method is implemented for conflict detection.

3. DroS foundations

This section is devoted to describing the different types of UASs, the airspace where their missions are performed, the equipment of the UAV, conflicts, and collisions, and the ontology Dronetology-cas (Martín-Lammerding, Astrain, & Córdoba, 2021) used in DroS.

3.1. UAS concepts and airspace environment

Unmanned Aircraft Systems (UASs) consist of the unmanned aerial vehicle (UAV), the equipment required to remotely control it (including the control link and the ground station), and the payload. A UAS can

Table 4
UAS categories with MTOM ranges (Watts, Ambrosia, & Hinkley, 2012).

	Nano	Micro	Mini	Light	Small	Tactical	MALE	HALE	Heavy	Super heavy
MTOM (kg)	0.2	2	20	50	150	600	1000	1000	2000	2500

be autonomous or remotely piloted (human-operated with a control system). The unmanned aircraft UAV's main parts are the auto-pilot, the flight control unit (FCU), multiple onboard sensors, and the communication equipment. Altimeters and Global Navigation Satellite Systems (GNSS) are the more widely used onboard sensors but not the only ones. Each UAV follows a unique identifier, is located at a given location (x, y, z), and has a given velocity $\vec{v}$ in an inertial system reference system at a given time instant t. Note that we refer to velocity as a vectorial representation of the movement, while speed is the module of this velocity.

A UAV flies over certain locations called waypoints, which are precise locations determined by GNSS. The aircrafts usually follow a flight plan ($FP^k_{u_i}$), a pre-flight plan indicating the totally ordered sequence of k waypoints that the aircraft u_i will follow during its flight and the expected navigation speeds between waypoints. Flight plans start at the take-off location, also called HOME, and end at the landing location (which may or may not coincide with HOME's location).

UAVs can be classified according to their Maximum Take-Off Mass (MTOM), the maximum mass at which a given UAV is certified, to their flight mode, and their landing/take-off mode. According to MTOM, ten different categories of UAV can be differentiated, as described in Table 4. MALE and HALE are the acronyms for medium altitude-long endurance and high altitude-long endurance, respectively. According to the flight mode, UAVs can be classified as copters (rotor wing), multicopters (multirotor), planes (fixed-wing), or hybrids (tiltrotors). Finally, according to the take-off and landing mode, UAS can be classified as vertical (VTOL), short (STOL) or conventional (CTOL) take-off and landing systems. Each UAS will belong to one class or another depending on its mass, mode of flight, and take-off and landing mode. The class of aircraft will imply a particular flight mode and maneuverability, which must be taken into account when dealing with the detection and avoidance of collisions since the aircraft features will limit evasive maneuvers.

The controlled airspace classes are defined in terms of flight rules and interactions between aircraft and Air Traffic Control (ATC). Most nations adhere to the classification specified by the International Civil Aviation Organization (ICAO). The ICAO airspaces allocate the responsibility for avoiding other aircraft, either to ATC (if separation is provided) or to the aircraft commander (if not). Very Low-Level Airspace (VLL) is the air volume below 500 ft above ground level (unbuilt). A flight Beyond Visual Line of Sigh (BVLOS) is the one in which the pilot has no direct visual contact with the UAS and no assistance from an observer.

Let the bearing in aeronautical navigation, denoted as $b_{u_i}(t)$, be the clockwise angle between the Magnetic North and the velocity vector of the UAV at a given time instant t. Let the relative bearing, denoted as $br(u_i, u_j, t)$, be the clockwise angle between the velocity vector $\overrightarrow{v_{u_i}}$ and the segment between u_i and u_j, at a given time instant t. Fig. 2 shows the bearing of each UAV and the relative bearings. The UAV telemetry is a time-series data that describes the state of the UAV at any time. In our case we use the UAV location, bearing and speed: $location_{u_i}(t)$, $b_{u_i}(t)$, $speed_{u_i}(t)$.

3.2. UAV equipment

The UAV equipment of a given UAV (u_i), Eq_{u_i}, is the set of equipment installed on u_i intended to detect any kind of aircraft traffic. This equipment can be of two types: collaborative or non-collaborative. Let a collaborative element, e_c, be a wireless device that allows bi-directional communication of air-traffic data between two UAV. A collaborative element (transponder) sends and receives air-traffic data periodically,

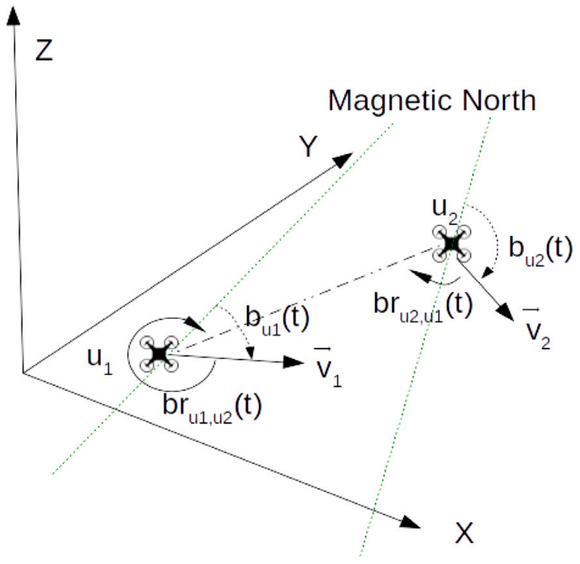


Fig. 2. Bearing $b_{u_1}(t)$ and $b_{u_2}(t)$ and relative bearing $br_{u_1u_2}(t)$ and $br_{u_2u_1}(t)$ of u_1 and u_2, at a given time t.

T_c, including the UAV identifier (id_{u_i}), the coordinates ($xyz(u_i, t)$), the UAV'speed ($speed(u_i, t)$), its bearing ($b(u_i, t)$), its class ($C^{u_i}_F$) and its flight type ($T^{u_i}_P$, plane or copter), at a given time t. After this, we will only consider traffic corresponding to UAV.

A collaborative UAV is one that communicates wirelessly with at least another UAV using a collaborative element in order to share knowledge about a common flight environment. The subset U_{ce} is the set of all UAVs collaborating at a given time in a given flight space. Likewise, a non-collaborative UAV is the one that lacks collaborative elements or, having them, does not share its knowledge of the airspace it flies over with any other UAV. Non-collaborative UAVs (U_{nce}) are divided into two sets, U_d, with those UASs that deploy any equipment that collects data about air traffic, and U_{nd}, with those UAVs without such equipment, such that $U_{nce} = U_d \cup U_{nd}$.

Each UAV may carry (or may not) a set of on-board sensors, s_z, to detect other UAV located in the same flight space. Onboard sensors collect data periodically, T_s, according to the sensing technology used. The most common data collected from sensors are the UAV identifications ($u_i \in U$), the euclidean distances between UAV ($d(u_i, u_j, t)$) and the UAV bearings at a given instant time.

3.3. Conflicts, obstacles and collisions

Let us define the protection volume of u_i, $V^{u_i}_P$, as the spherical moving volume centered at its location during the flight plan $FP^k_{u_i}$ that must be clear from any other UAV in order to grant the proper completion of the mission as planned. A conflict occurs whenever a moving object flying in $V^{u_i}_P$ could collide with a given UAV. A *conflict* between u_i and u_j occurs if the distance between both UAVs is lower than the radius r_p of the protection volume $V^{u_i}_P$. For our purposes, every conflict detected by a given UAV u_i is caused by another UAV u_j. A collision between u_i and u_j ($u_i \neq u_j$) is a physical contact between them that unables both to continue their flight plans. We define the spherical collision volume $V^{u_i}_{CC}$, with radius r_{CC}, as the largest spherical volume around u_i where any UAV inside it will collide against each other due to the inertia of the UAV. A collision occurs between two UAV u_i and u_j

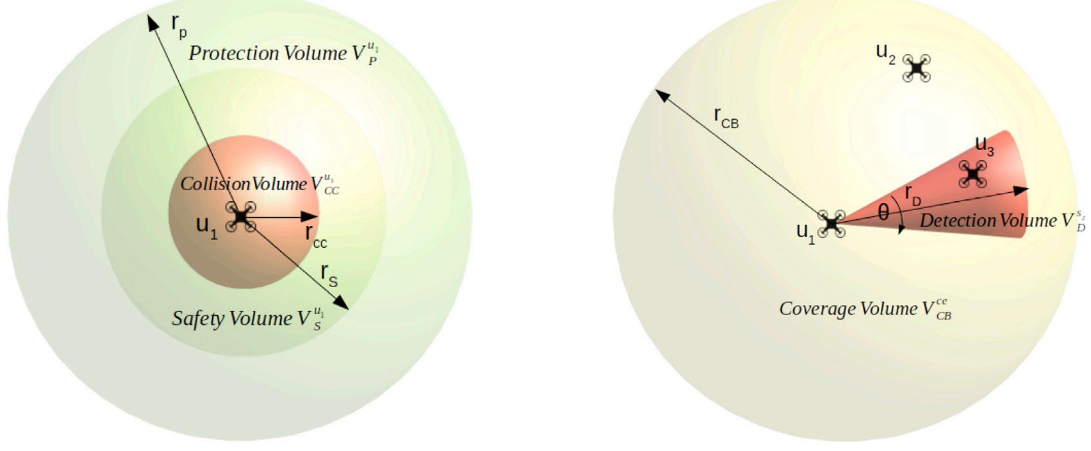

r_p
Protection Volume $V_P^{u_1}$
Collision Volume $V_{CC}^{u_1}$
u_1
r_{cc}
r_S
Safety Volume $V_S^{u_1}$
u_2
r_{CB}
u_3
r_D
θ
Detection Volume $V_D^{s_z}$
u_1
Coverage Volume V_{CB}^{ce}

Fig. 3. Protection, safety and collision volumes (left) and detection and coverage volumes (right).

when the euclidean distance between them $d(u_i, u_j, t) \le r_{CC}$. The safety volume $V_S^{u_i}$ is defined as the smallest spherical moving volume centered at the location of u_i during its flight plan that must be clear from any other UAV in order to avoid a collision, being r_S the radius of $V_S^{u_i}$.

A Collision Avoidance System (CAS) is a safety element devoted to prevent collisions, ensuring that UAVs do not enter the safety area of other UAVs, and if they do, that they leave it immediately without impacting or threatening the safety of their flight ($d(u_i, u_j, t) > r_{CC}, \forall t, \forall u_i, u_j \in U$).

We define the detection volume $V_D^{s_z}(u_i)$ centered at the u_i location, with radius r_D and a solid angle θ, where the sensibility of the sensor s_z allows to detect other UAVs. The coverage volume of a collaborative element $V_C^{ce}(u_i)$ is the sphere of radius r_C and centered at the location of u_i where the signal-to-noise ratio (SNR) is enough to grant the correct reception of the data broadcasted by the transponder. A given u_i discovers u_j when it receives some data from u_j or whenever any of its sensors detect the presence of u_j. Note that a conflict may exist between two UAVs even if one has not discovered the other or they have not discovered each other.

As a summary, Fig. 3 (left) shows $V_P^{u_i}$, $V_S^{u_i}$ and $V_{CC}^{u_i}$ for a given UAV u_i. Different values of r_P, r_S and r_{CC} are defined for each UAV class C and type T. Fig. 3 (right) shows $V_D^{s_z}$ and V_C^{ce} for u_1 equipped with a collaborative element ce and a sensor s_z. The UAV u_2 is located in $V_D^{s_z}$ while u_3 is located in $V_D^{s_z}$ and in V_C^{ce}. In order to deal with conflicts, we define U_C as the subset of conflicting UAVs ($U_C \subset U$) at a given time t, such that the distance between each pair of UAVs is lower than the protection radius (r_P). In the same way, we define $U_C^{u_i}$ as the subset of conflicting UAVs with a given UAV u_i. During an UAV flight, the elements of the subset $U_C^{u_i}$ may vary during the time due to changes in the distance between pairs of UAV, their bearing, their location, sensor failures, or radio interference. We define V_C as the smallest volume that contains every $u_i \in U_C$, being the number of conflicts n_C the cardinal of U_C ($n_C = |U_C|$).

3.4. DroS's ontology: Dronetology-cas

As described above, DroS is based on the ontology Dronetology-cas, available at Martín-Lammerding, Astrain, and Córdoba (2022b), which is an adaptation of Dronetology (Martín-Lammerding, Astrain, & Córdoba, 2022c). According to the level of generalization defined in Guarino (1998), Dronetology-cas is an application ontology intended as a universal vocabulary for any CAS for UAVs. We include a short description of Dronetology-cas so that this article is self-contained in order to improve its readability. The ontology has been implemented using Stanford University (2022) and the Web Ontology Language (OWL) (Hitzler, Krötzsch, Parsia, Patel-Schneider, & Rudolph, 2009).

Dronetology-cas defines a knowledge base (KB) that stores flight and conflict data using RDF (Lassila & Swick, 1999). The basic unit of RDF is a triplet, consisting of a subject, a predicate, and an object. Note that a KB consists of an ontology and the set of individual instances of its classes (Noy & McGuinness, 2001) and (Guarino & Giaretta, 1995). The Dronetology-cas KB also stores the knowledge, in terms of triplets, extracted from local sensors and remote UAS, the inferred knowledge about conflicts and the avoidance maneuvers. Dronetology-cas elements, as defined in the OWL specification, are identified using International Resource Identifiers (IRIs). The abbreviated syntax is *drone:localname*, where *drone:* refers to the Namespace http://dronetology.net/dronetology-cas#. *localname* is the name of the class, relation, attribute or individual defined in Dronetology-cas.

The knowledge stored in the KB can be classified as spatial and time-dependent. Spatial knowledge requires storing positions of the individuals in the form of coordinates, that are defined with the attributes latitude, longitude, and altitude. These attributes are the UTM projection relative to the World Geodetic System 1984 (WGS84) coordinate system. The altitude of conflicts is relative to the Mean Sea-Level (MSL). To improve information interoperability, the positions defined use the *geo:wktLiteral* data type with a WGS 84 geodetic latitude-longitude.

Time-dependent knowledge is modeled using time-series data. Time-series data is a sequence of data collected over time intervals, allowing for tracking changes of a certain magnitude over time. KB stores time-series data concerning conflicts and the local UAV u_i for inference and accountability.

Instances of classes defined in Dronetology-cas that stores time-series data relate with an Iteration instance, that represents the number of iterations elapsed at a given time, using the object property *drone:belongsTo*.

The Dronetology-cas design considers the computational limitations of onboard systems. Thus, memory usage has been reduced by limiting the number of classes in the model and avoiding importing auxiliary ontologies. The main classes of Dronetology-cas are *UAS*, *MissionElement*, *InputData*, *AntiCollisionSystem* and *Conflict*. MissionElement and AntiCollisionSystem are defined as super-classes of a set of related classes that detail parts of them. This relationship is defined as a grouping rather than a specialization, as it simplifies the definition of relations and allows sharing of attributes. Fig. 4 depicts the main classes of Dronetology-cas.

A *Conflict* is caused by a remote UAS and detected by the UAS' equipment. The *cause* relation between remote UASs and conflicts is modeled as a sub-type, i.e. *Conflict* is a subtype of *UAS*. Therefore, a *Conflict* inherits the dynamic attributes of the remote *UAS* that caused it. Consequently, if multiple UASs are in a mutual conflict, Dronetology-cas models this situation with multiple conflicts.

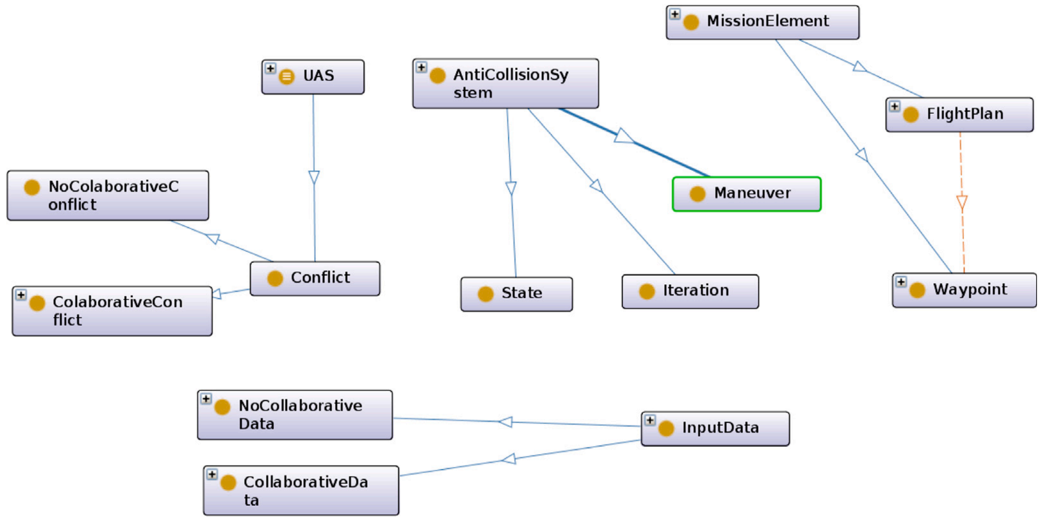


Fig. 4. Dronetology-cas main classes.

The *MissionElement* class groups all the elements of a mission. Both *Waypoint* and *FlightPlan* classes derive from *MissionElement*. The *InputData* class represents any data collected from external sources (a sensor for UAS detection, a collaborative element, the GNSS, or the FCU). Data collected from sensors are modeled using the class *NoCollaborativeData*, a sub-class of *InputData*. Subclasses of *NoCollaborativeData*, such as *VisionData*, *LidarData* or *RadarData*, define independent data sources that are non-collaborative. Data received from the collaborative element is represented using the class *CollaborativeData*, a sub-class of *InputData*. Subclasses of *CollaborativeData* are *ADSBData* and *FLARMData*. *FCUData* and *GNSSData* are subclasses of *InputData*, that represent the UAS flight status and its position, respectively.

Any instance of *NoCollaborativeData* or *CollaborativeData* identifies a conflict. Dronetology-cas also classify conflicts according to their collaborative capability. For this purpose, the subclasses *NoCollaborativeConflict* and *CollaborativeConflict* are defined. The object property *drone:detect* relates *NoCollaborativeData* or *CollaborativeData* individuals with *Conflict* individuals. An individual of type *Conflict* has the following attributes: latitude, longitude, altitude, bearing, speed and relative bearing. These attributes are initialized from the related *InputData* individuals. A *Conflict* class instance has a conflictID attribute, regardless of the conflict data source.

The class *DroSConflict* is defined as a subclass of *CollaborativeConflict* to differentiate conflicts caused by DroS-equipped UAS from other *CollaborativeConflicts*. The *Conflict* attribute *fcd* is defined to store the result of a collision detection function, $f_{cd}(U_C^{u_i}, t)$. Another Conflict attributes are *mtom* and *ttc* devoted to store MTOM and time to collide respectively. The attributes *coordinatedPriority* and *coordinatedSpeed* of class *CollaborativeConflict* store the priority and the coordinated speed when the UAS is in state *Coordinated*.

The *AntiCollisionSystem* class groups CAS elements. In our model, CAS has a state, may have an avoidance maneuver, and executes continuously repeating iterations. The *State*, *Maneuver* and *Iteration* classes derive from the class *AntiCollisionSystem*. Next position of the local UAS, NextIterationLocalUASLocation, and conflict position predictions, NextIterationConflictLocation, are also subclasses of *AntiCollisionSystem*. *NextIterationLocation* class is derived to group every position in the next iteration. Instances of classes that stores time-series data are *InputData*, *Conflict*, *NextIterationConflictLocation*, *NextIterationLocalUASLocation* and *State*. Instances of the *Iteration* class relates instances of different classes that exists in the KB at each iteration time interval, using the object property *drone:atIteration*.

The state of the CAS, at a given iteration, is represented as an instance of the *State* class with an attribute that encodes the state,

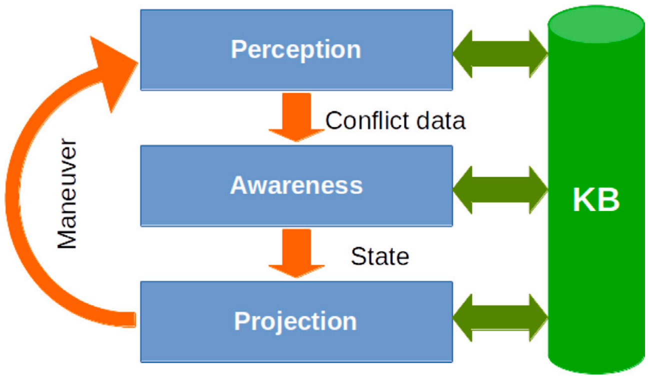


Fig. 5. DroS autonomous anti-collision system layers.

stateCode, and a relation *drone:atIteration*, related with *drone:belongsTo* by *owl:inverseOf*, with the corresponding *Iteration* instance.

The *Maneuver* class represents if the planned FP has been changed at a given time. Two subclasses, *SpeedManeuver* and *BearingManeuver*, are defined to accommodate conflict avoidance maneuvers.

4. DroS's inference for collision avoidance

Knowledge about conflicts are used for the conflict avoidance decision-making process. DroS executes onboard inference to avoid conflicts collaborating with other UAVs. DroS's inference consists of three layers, the *perception* layer, the *awareness* layer and the *projection* layer, previously applied in other CAS and command and control systems (Endsley, 1995; Feng, Teng, & Tan, 2009). Knowledge inference improves the situational awareness of DroS and the conflict avoidance capabilities, as summarized in Table 5.

The layered and modular design of DroS's inference facilitates the adaptation to different sensor systems, collision detection techniques, or collision path avoidance calculation methods. Each layer executes sequentially and continuously, as depicted in Fig. 5, inferring and adding new knowledge to the KB during the UAS flight. The insertion of data into the KB and the transmission of an FCU command are executed independently of the three layers of DroS.

DroS has a KB that consists of Dronetology-cas and a set of individuals. Data stored in the KB is modeled using RDF. For example, the sentence "the remote UAS identified by Id2 has a collaborative element"

Table 5
DroS inference capabilities summary.

Inferred knowledge	Inference
Real conflicts	As multiple conflict data sources may lead to duplicate conflicts, inference avoids unnecessary maneuvers and performs a more precise maneuver
Conflict's attributes	As some conflicts attributes may not be available from sensors, inference assigns a value considering the log history
Conflict's position	As sensor or collaborative elements may not have updated data from conflicts, inference locates the conflict considering the log history
Conflict type	As DroS can be introduced in any kind of traffic, inference classifies conflict types in order to select a conflict avoidance method
Conflict avoidance method	As DroS combines two avoidance methods, inference selects the most suitable knowing conflicts characteristics
Sensor diagnosis	As multiple conflict data sources are available in DroS, inference detects if one has failures comparing values with other

is modeled in RDF as *remoteUASId2* (subject) *has* (predicate) *collaborativeElement* (object). In Turtle format (Beckett & Berners-Lee, 2008), the above sentence is expressed as:

```
drone:remoteUASId2   drone:hasPart   drone:collaborativeElement
```

Asserted triplets are initially defined or are inserted in the KB after the transformation of data from sensors or telemetry into triplets. Inferred triplets are additional triplets created by an inference engine that applies logical rules to the knowledge base (asserted or inferred) to deduce new triplets. Therefore, inferred triplets are new knowledge available in the KB.

Knowledge is generated in DroS' KB through the inference that is defined using SPARQL queries (Prud'hommeaux & Seaborne, 2008). New knowledge is obtained by executing SPARQL queries that create new triplets or return a specific answer.

When addressing autonomy, DroS inference requires the definition of rules that extend the inference capabilities of OWL and RDF. Rules are implemented in SPARQL, following the SPIN (Knublauch, Hendler, & Idehen, 2011) standard. Certain anti-collision functions are not easily translated to SPARQL so we implement them as ARQ functions (Apache Foundation, 2022b). We review for each DroS layer the main SPARQL queries designed for collision avoidance.

4.1. Perception layer

The *perception* layer is in charge of preparing the KB before starting a new Iteration instance, creating an instance of the *Iteration* class in the KB to initialize a CAS iteration. The *drone:belongsTo* object-property relates all class instances with time-series data with its corresponding instance of Iteration.

The *perception* layer estimates conflict's attributes using past values. It also improves situational awareness as it can reload conflicts of past iterations that are not detected in the current iteration.

Listing 1 depicts the SPARQL query used to reload conflicts of past iterations that are not discovered at the current iteration, but should be considered in a collision avoidance maneuver.

The above SPARQL starts searching the highest iteration number obtained from the *SELECT* sentence at line 15. In the next step, the *SELECT* at line 8 obtains the last Iteration instance using the highest iteration number. This instance is used to find conflicts related to the object property *drone:belongsTo* in the last 20 iterations (lines 23–24). The number of past iterations considered (20) is a configuration parameter that depends on the duty cycle of the sensor that detects conflicts.

4.2. Awareness layer

The *awareness* layer fuses data from different sources using the homogeneous data provided by the *perception* layer. It also updates the

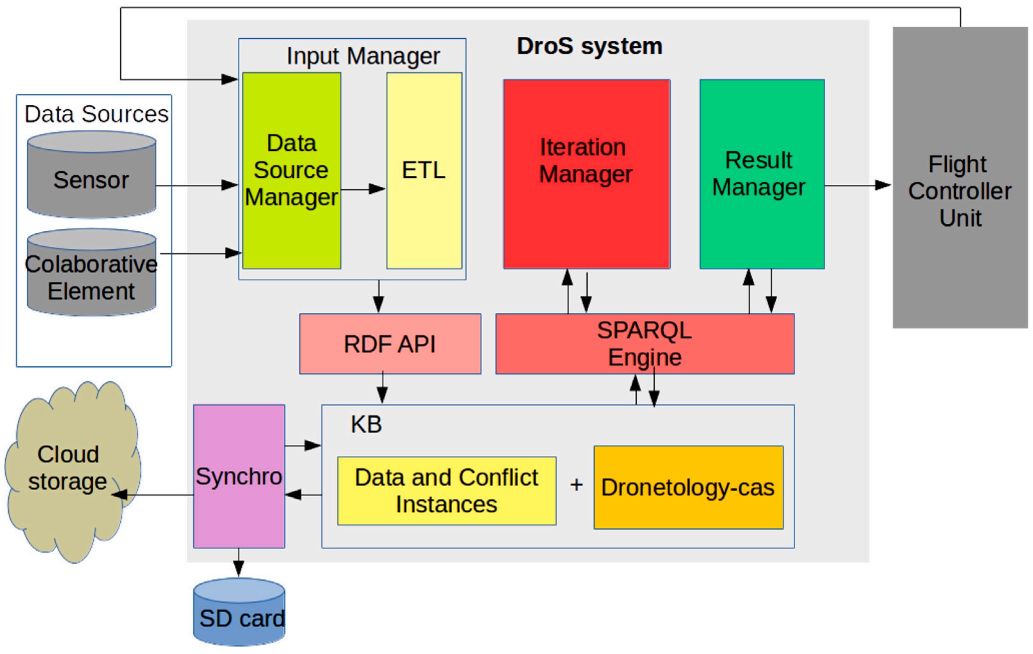


Fig. 6. $DroS_{u_i}$ building blocks.

Listing 1: SPARQL query to identify recent conflicts not discovered in the current iteration.

```
SELECT {
        ?conflict.
    }
WHERE {
    ?conflict a drone:Conflict.
    ?conflict drone:belongsTo ?conflictIteration.

   SELECT ?iteration
   WHERE
   {
                ?iteration a drone:Iteration.
                ?iteration drone:hasIterationSequence ?iterationSequence.
                FILTER (?iterationSequence = ?lastIterationSequence)
                {
                    SELECT (MAX(?num) AS ?lastIterationSequence)
                    WHERE
                    {
                        ?iterationTemp a drone:Iteration.
                        ?iterationTemp drone:hasIterationSequence ?num.
                    }
                }
    }
    FILTER( abs(?iteration-?conflictIteration)>1 ).
    FILTER( abs(?iteration-?conflictIteration)<20 ).
}
```

individual of class *State* at each iteration and classifies collaborative UAS. The discovery of a remote UAS by multiple data sources may generate duplicate conflicts. To avoid this, the *awareness* layer implements a data fusion process. The data fusion process adds to the KB the object property *drone:fusedWith* to relate individuals from different *InputData* sub-classes which refer to the same conflict. The criteria used to identify data about the same conflict are proximity, relative bearing, and data collected at the same given time interval.

Listing 2 depicts the SPARQL query used to create the relationship *drone:fusedWith* between an individual of the class *ADSBData* and another of *VisionData* class when the relative bearing difference is lower than 20 degrees, the distance between them is lower than 15 m. and both data instances are collected in the same iteration.

```
CONSTRUCT {
     ?adsb drone:fusedWith ?vision.
}
WHERE {
    ?adsb a drone:ADSBData.
    ?vision a drone:VisionData.
    ?adsb drone:hasRelativeBearing ?bear1.
    ?vision drone:hasRelativeBearing ?bear2.
    ?adsb drone:belongsTo ?iter1.
    ?iter1 drone:hasIterationSequence ?iterId1.
    ?vision drone:belongsTo ?iter2.
    ?iter2 drone:hasIterationSequence ?iterId2.
    ?adsb drone:hasDistance ?distance1.
    ?vision drone:hasDistance ?distance2.
    FILTER( abs(?bear1-?bear2)<20 ).
    FILTER( abs(?distance1-?distance2)<15 ).
    FILTER( iterId2=?iterId1 ).
}
```

Listing 2: SPARQL query to fuse data based in relative bearing and proximity.

Once the *drone:fusedWith* relations of the data fusion process are inserted, the *awareness* layer creates *Conflict* individuals from instances of type *InputData* depending on whether or not they have an *drone:fusedWith* relationships. The relationship between an individual of type *InputData* and an individual of type *Conflict* is represented in the KB with the object property *drone:isConflict*, equivalent to *drone:detect*, that the *awareness* layer inserts.

Another inference of the *awareness* layer is the one that classifies remote UAVs and initializes the state of the CAS between idle, collaborative, or non-collaborative.

The *awareness* layer adds knowledge to the conflicts stored in the KB, initializing *fcd*, *mtom* and *ttc data properties* of a *Conflict* instance. It also updates the state of the DroS system by checking the type of individuals of the super-class *Conflict* at the current iteration. Whenever it detects conflicts in the protection sphere, the *awareness* layer changes the CAS state to an active state (collaborative or non-collaborative).

4.3. Projection layer

The *projection* layer selects the avoidance method to be applied, depending on the updated state stored by the *awareness* layer, and stores the resulting maneuver, obtained from the avoidance methods, into the KB. When CAS is active, the *projection* creates an individual of the *Maneuver* class initialized with the avoidance method selected. As a conflict avoidance maneuver lasts more than an iteration, multiple *Maneuver* instances exist. To group them, the *projection* layer uses the object-property *drone:sameManeuver*.

The three-layer structure of DroS are explained with the following example. It shows the DroS' KB of one of the UAVs, denoted as u_1, involved in a conflict. Instances names are constructed using the iteration number to facilitate understanding. The perception layer starts a new iteration, as an example, the 23rd, so the following triplets are asserted in KB_{u_1}:

```
drone:iter-23   owl:Class   drone:Iteration
drone:iter-23 drone:hasIterationSequence 23^^xsd:int
```

u_2 (identified by "idu2") sends an ADS-B message that u_1 receives at iteration number 23. In the same iteration, the vision camera of u_1 detects a conflict, which is identified by a code generated using a consecutive index, for example, "vc32". The triplets asserted are:

```
drone:adsb-idu2-iter-23   owl:Class   drone:ADSBInputData
drone:vc-vc32-iter-23   owl:Class   drone:VisionInputData
```

The awareness layer fused the above triplets, as both detected conflict are located near. As a result, the individuals of *InputData* are related as both identify the same conflict.

```
drone:vc-vc32-iter-23   drone:fusedWith
 drone:adsb-idu2-iter-23
```

Therefore, there is only a collaborative conflict, as the remote UAS broadcasts ADS-B messages. This implies that DroS changes its state to "coordinated". The triplets inferred from the previous are:

```
drone:conflict-idu2-iter-23   owl:Class
 drone:CollaborativeConflict
drone:state-iter-23   owl:Class   drone:State
drone:state-iter-23   drone:hasState
 "coordinated"^^xsd:string
```

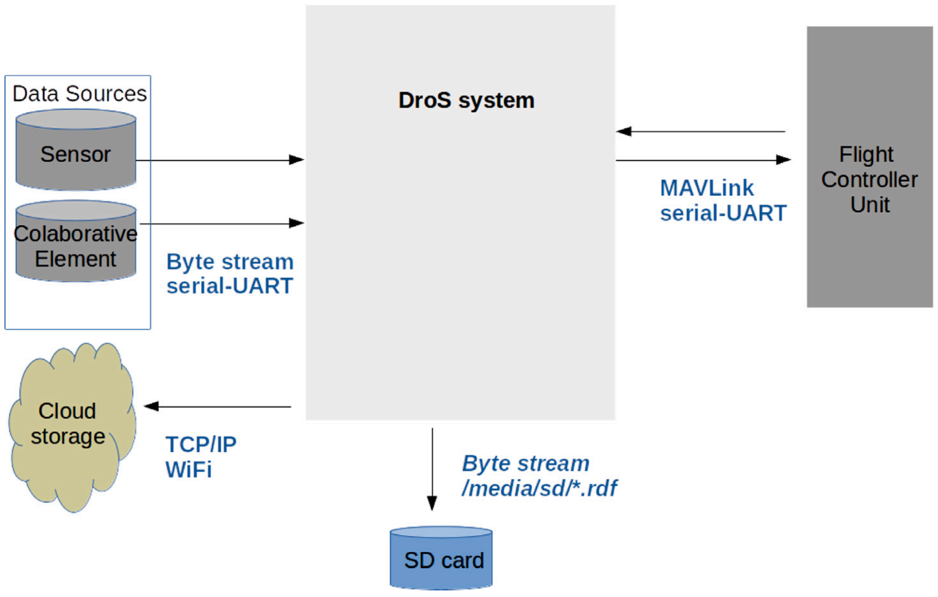


Fig. 7. $DroS_{u_i}$ interfaces.

As the conflict is collaborative and the function f_{cd} is true, DroS infers at the projection layer an avoidance maneuver based on adapting UAVs speeds.

```
drone:maneuver-iter-23   owl:Class   drone:SpeedManeuver
```

If the conflict persists in the next iteration, DroS will add triplets to the KB like the following:

```
drone:iter-24   owl:Class   drone:Iteration
drone:iter-24    drone:hasIterationSequence   24^^xsd:int
drone:conflict-idu2-iter-24   owl:Class
 drone:CollaborativeConflict
drone:state-iter-24   owl:Class   drone:State
drone:state-iter-24   drone:hasState
 "coordinated"^^xsd:string
drone:maneuver-iter-24   owl:Class   drone:SpeedManeuver
drone:maneuver-iter-24   drone:sameManeuver
 drone:maneuver-iter-23
```

5. DroS's system implementation

DroS architecture is depicted in Fig. 6. $DroS_{u_i}$ runs continuously, over u_i, fusing data collected from sensors, updating the knowledge about conflicts, and estimating the future locations of the remaining conflicting UAV in an infinite loop with a minimum period of T_{iter}. A DroS iteration is a complete execution of this loop. Iterations are numbered sequentially, starting in 1 until the flight plan ($FP^k_{u_i}$) is over. KB_{u_i} is a repository of knowledge concerning the subset of conflicting UAVs with u_i ($U^{u_i}_C$). A DroS-equipped UAV can be configured, for example, with an ADS-B transceiver, a camera-based sensor, and an onboard microcomputer. ADS-B is a well-known collaborative air-traffic technology, and camera-based sensors are widely used to detect conflicts in UAVs.

DroS implements multiple interfaces to retrieve data from detected conflicts and to perform maneuvers to avoid them. Connections to UAS components require different protocols, as detailed in Fig. 7. DroS connects with the FCU via MAVLink protocol (Koubaa et al., 2019) as it is a defacto standard protocol for this purpose. It allows to read telemetry data and send commands to the FCU using MAVLink protocol. Sensors and transponders are connected using a serial-UART connection transferring a byte stream. The KB is serialized in a file that is saved to an SD Card. A cloud storage provider allows saving the file with the KB using Wifi and TCP/IP protocols.

In order to detect possible collisions, we define a boolean time-dependent collision detection function $f_{cd}(u_i, U^{u_i}_C, t)$ that assess if a remote UAVs $u_j \in U^{u_i}_C$ might collide with u_i at a given instant time t. An example of this kind of function may be the Constant bearing-decreasing range (CBDR) one. Let $t_{tc}(u_i, u_j)$ be the worst-case time to a collision between two UAVs u_i and u_j the quotient between the euclidean distance between this pair of UAVs and the relative speed between them. In the same way, let t_{avo} be the avoidance time before a collision occurs between u_i and u_j at a given instant t. The value of t_{avo} is determined by the quotient between the safety radius of u_i and the relative speed between u_i and u_j at a given instant time t ($t_{avo}(u_i, u_j, t) = \frac{r^{u_i}_S}{speed_{relative}(u_i,u_j,t)}$).

$DroS_{u_i}$ collects data from sensors, collaborative elements and the FCU with the *Input Manager*. The *Input Manager* collects data according to the period of each data source, T_s or T_c, and inserts *InputData* triplets in the KB, creating individuals through the RDF API. The *Input Manager* is composed by the *Data Sources Manager* and the *ETL* (Extract, Transform, and Load) component, which provides uniformity and interoperability to data. The *Data Sources Manager* is an abstraction layer that allows DroS to interact with external data sources, such as sensors, collaborative elements, and the FCU. The *Input Manager* implements the connection with any external data source and determines which data sources are active and what kind of connection is required. The *ETL* performs the function of polling data from the *Data Source Manager*, transforming the incoming data into triplets and inserting them in the KB.

KB_{u_i} is composed of Dronetology-cas and state-dependent data about conflicts and $DroS_{u_i}$'s state. The KB_{u_i} stores knowledge from $DroS_{u_i}$'s operation as a *fligh data recorder* does. $DroS_{u_i}$ is implemented with the Apache Jena framework (Apache Foundation, 2022a) and Java 8. Apache Jena implements an RDF API and a SPARQL Engine.

$DroS_{u_i}$ runs on an embedded computer with limited main memory and a storage card (SD) as secondary memory. As the UAS fly, the KB_{u_i} grows and the $DroS_{u_i}$ response time increases (Martín-Lammerding et al., 2021). In order to minimize the main memory usage and reduce the search time, only a partial KB_{u_i} with n_{iter} past iterations is kept in the main memory. The KB_{u_i} is stored in the SD. Asynchronously and simultaneously, the *Synchro* module keeps knowledge from the last n_{iter} iterations in the KB_{u_i}, stores the complete KB_{u_i} in the SD and, if enabled, copies the KB_{u_i} to a cloud storage.

Table 6
DroS inference related to the avoidance method and the DroS's layer.

Inferred knowledge	Avoidance method	DroS's layer
Real conflicts	CS+EA	Perception
Conflict's attributes	EA	Awareness
Conflict's position	CS+EA	Projection
Conflict type	CS	Projection
Conflict avoidance method	CS+EA	Awareness
Sensor diagnosis	CS+EA	Perception

The *Iteration Manager* coordinates the execution of the three layers of the inference. It uses the *sparql-engine* component to create new individuals and to query the KB_{u_i} for specific knowledge. The *Result Manager* retrieves a conflict avoidance decision from the KB_{u_i} through SPARQL queries and transforms it to an FCU command, which is executed with the highest priority.

6. DroS conflict avoidance

DroS conflict avoidance is based on inferring knowledge to avoid collisions. Knowledge management allows selecting the most efficient CAS method in terms of flight time and fidelity to the aircraft's initial flight plan, minimizing possible bearing corrections and delays. Knowledge can be used in many ways to avoid conflicts. However, in this paper, we create two novel methods: Coordinated Speed (CS) and Evasive Action (EA). DroS avoids conflicts by executing the CS method on those UAVs equipped with collaborative DroS, while the EA method can be applied in any scenario.

DroS follows the CS method to adapt the speed in a collaborative way when there is enough time margin ($t_{tc}(u_i, u_j, t) > t_{avo}(u_i, u_j, t)$). When this margin does not exist ($t_{tc}(u_i, u_j, t) < t_{avo}(u_i, u_j, t)$), DroS changes the UAV bearing using the EA method. CS allows to accomplish both objectives, but its application is limited to DroS-equipped UAVs and requires knowledge about the conflicts. In contrast, the EA can be used in any case and avoids collision regardless of the equipment of the UAV in conflict, although it may change its flight plan.

Let t_m be the maximum time that may elapse until the accomplishment of CS. Note that after this period, the speed adjustment of the UAVs involved in a given conflict can be ensured. Therefore, a DroS-equipped UAV may infer which remote UAVs have applied coordinated maneuvers due to the reception of the remote UAVs' speed through the collaborative elements.

DroS can reach different states depending on the method chosen. DroS reaches the *Coordinated* state (collaborative CAS state) when CS is active, the *Evasive* (non-collaborative CAS state) when an EA is active, or *Planned* (idle CAS state) when no conflicts occur. CS requires a collaborative element, like ADS-B, and EA requires a non-collaborative sensor for conflict detection, like a video camera.

As a conflict avoidance summary, Table 6 depicts DroS's inference related to the avoidance method and the DroS's layer that implements it.

6.1. DroS's coordinated speed (CS)

CS_{u_i} modifies the speed of a given DroS-equipped UAV u_i to avoid collisions with other UAVs located in the same scenario. For this purpose, a total order relation is established on the set $U_C^{u_i} \cup \{u_i\}$. The ordered set is obtained by sorting the UAV by its identifiers, id_{u_i}. The new speed is obtained from the position occupied by u_i in the set of target speeds S_p. Let U_{dros} be the subset of DroS-equipped UAS on the flight space. CS_{u_i} is applied in a given detection volume (V_D). Target speeds are defined as follows: $S_p = \{sp_o\} \quad o : 1..n_C, o \in \mathbb{N}, |\forall sp_o \in S_p, sp_o \geq sp_{o+1}$. Conflict instances in the ordered set are obtained from the KB using SPARQL sentences. Pseudocode 1 shows the CS method.

Pseudocode 1 . $\mathrm{CS}_{\mathrm{u_i}}$ method

```
 1: Being at time t, $\forall \mathrm{u_i} \in \mathrm{U_{dros}}$
 2: while $| U_C^{u_i} | > 0 \wedge U_C^{u_i} \subset U_{ce} \wedge \forall \mathrm{u_k} \in U_C^{u_i} \mid \mathrm{t_{tc}(u_i,u_k)} > \mathrm{t_{avo}} \wedge \exists \{\mathrm{u_j}\} \in U_C^{u_i} \mid \mathrm{f_{cd}(u_i, u_j, t)}$=true do
 3:     $U_{Cu_i}^{u_i} \leftarrow U_C^{u_i} \cup \{u_i\}$
 4:     $U_{COu_i}^{u_i} \leftarrow U_{Cu_i}^{u_i} \mid id_{u_k} > id_{u_{k+1}}, \forall u_k \in U_{Cu_i}^{u_i}$
 5:     for all $u_k \in U_{COu_i}^{u_i}$ do
 6:         if $u_i = u_k$ then
 7:             $pos_{u_i} \leftarrow indexOf(U_{COu_i}^{u_i}, u_i)$
 8:         end if
 9:     end for
10:     $sp_{u_i}(t) \leftarrow elementAt(Sp, pos_{u_i})$
11:     $FCU \leftarrow sp_{u_i}(t)$
12:     $KB \leftarrow U_{COu_i}^{u_i}, pos_{u_i}, sp_{u_i}(t)$
13: end while
```

CS_{u_i} obtains a set of conflicting UAVs (line 2) that includes the local UAV, $U_{Cu_i}^{u_i}$. CS_{u_i} obtains the ordered set $U_{COu_i}^{u_i}$ by id_{u_i} (line 4). The UAV's id is used to sort the set of the discovered collaborative conflicting UAVs and the local UAV. CS_{u_i} finds the position of the local UAV in $U_{COu_i}^{u_i}$ and obtains the speed located at the same position of the set S_p. Once executed the assignation, CS_{u_i} changes the speed of the local UAV to the one assigned. Once CS_{u_i} obtains the speed (line 10); it indicates this speed to the controller (line 11) and updates the KB with the new coordinated maneuver (line 12). As DroS shares its flight data with others using a collaborative element, every DroS-equipped UAV in V_D knows the speeds of the remote collaborative conflicting UAVs and whatever speed change is performed. As DroS knows the assigned target speeds for the remote UAVs, it can check the application of the coordinated assignation. CS_{u_i} must be re-evaluated in every iteration since the set of discovered conflicting collaborative UAVs varies in time and from one UAV to another.

6.2. DroS's evasive action (EA)

EA_{u_i} modifies the bearing when at least one conflicting UAV with a time to collision shorter than the avoidance time, $t_{tc}(u_i, u_j, t) < t_{avo}(u_i, u_j, t)$, is detected. An EA avoids conflict by applying the PF technique to obtain a maneuver. In each DroS iteration, the PF implementation, derived from Zhao et al. (2017), provides a new location for the current UAV based on the initial position of the current UAV and the locations of the conflicting UAVs. Conflict instances that meet the EA method condition are obtained from the KB using SPARQL sentences. Pseudocode 2 depicts EA avoidance method including PF. The repulsion force that u_k applies over u_i is obtained from the relative speed and the distance between them (line 4).

The summation of repulsion forces from every conflicting UAV (line 6) results in a vector $\vec{force}_{u_i}(t)$ that provides a new bearing $b_{u_i}(t)$ (line 7). Finally, the new bearing is sent to the FCU, and results from every step are stored in the KB.

Pseudocode 2 . $\mathrm{EA}_{\mathrm{u_i}}$ method

```
 1: Being at time t, $\forall \mathrm{u_i} \in \mathrm{U_{dros}}$
 2: while $| U_C^{u_i} | > 0 \wedge \exists \{u_k\} \in U_C^{u_i} \mid \mathrm{t_{tc}(u_i,u_k)} <= \mathrm{t_{avo}} \wedge \mathrm{f_{cd}(u_i, u_k, t)}$=true
    do
 3:     for all $u_k \in U_C^{u_i}$ do
 4:         $\vec{force}(u_i, u_k, t) \leftarrow (\vec{v}_{relative}(u_k, u_i, t), distance(u_k, u_i, t))$
 5:     end for
 6:     $\vec{force}_{u_i}(t) \leftarrow sum(\vec{force}(u_i, u_k, t))$
 7:     $b_{u_i}(t) \leftarrow \vec{force}_{u_i}(t)$
 8:     $FCU \leftarrow b_{u_i}(t)$
 9:     $KB \leftarrow b_{u_i}(t), sp_{u_i}(t)$
10: end while
```

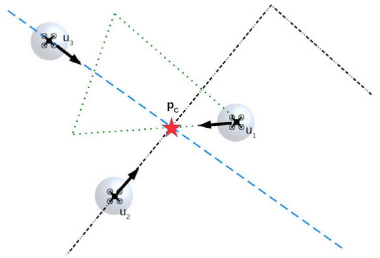
$\boxed{n_C\text{=3 at time } t_0}$

CS configuration:

$n_C^{max} = 5$

$S_p = \{25, 20, 15, 10, 5\}$

$U_{ec}=\{u_1,u_2,u_3\}$



sp_{u_1}=22 m/s	sp_{u_2}=23 m/s	sp_{u_3}=24 m/s
id_{u_1}=1	id_{u_2}=2	id_{u_3}=3
b_{u_1}=260°	b_{u_2}=45°	b_{u_3}=135°
$Eq_{u_1}=\{e_c, \text{DroS}\}$	$Eq_{u_2}=\{e_c, \text{DroS}\}$	$Eq_{u_3}=\{e_c, \text{DroS}\}$
$\text{d}(u_1,u_2)< r_C$	$\text{d}(u_2,u_3)< r_C$	$\text{d}(u_3,u_1)< r_C$
$\text{d}(u_1,u_3)< r_C$	$\text{d}(u_2,u_1)< r_C$	$\text{d}(u_3,u_2)< r_C$
$t_{tc}(u_1,u_2)$=18s	$t_{tc}(u_2,u_3)$=20s	$t_{tc}(u_3,u_1)$=22s
$t_{tc}(u_1,u_3)$=22s	$t_{tc}(u_2,u_1)$=18s	$t_{tc}(u_3,u_2)$=20s
$t_{avo}(u_1,u_2)$=15s	$t_{avo}(u_2,u_3)$=18s	$t_{avo}(u_3,u_1)$=16s
$t_{avo}(u_1,u_3)$=16s	$t_{avo}(u_2,u_1)$=15s	$t_{avo}(u_3,u_2)$=18s
$U_{ec}^{u_1}=\{u_2,u_3\}$	$U_{ec}^{u_2}=\{u_1,u_3\}$	$U_{ec}^{u_3}=\{u_2,u_3\}$
$U_C^{u_1}=\{u_2,u_3\}$	$U_C^{u_2}=\{u_1,u_3\}$	$U_C^{u_3}=\{u_2,u_3\}$
$\lvert U_C^{u1}\rvert = 2$	$\lvert U_C^{u2}\rvert = 2$	$\lvert U_C^{u3}\rvert = 2$
$f_{cd}(u_1,u_2,\text{t})$=true	$f_{cd}(u_2,u_3,\text{t})$=true	$f_{cd}(u_3,u_1,\text{t})$=true
$f_{cd}(u_1,u_3,\text{t})$=true	$f_{cd}(u_2,u_1,\text{t})$=true	$f_{cd}(u_3,u_2,\text{t})$=true
$\lvert U_C^{u_i} \rvert > 0 \land U_C^{u_i} \subset \text{U}_{\text{ec}} \land \forall\ \text{u}_\text{k} \in U_C^{u_i} \mid \text{t}_{\text{tc}}(\text{u}_\text{i},\text{u}_\text{k}) > \text{t}_{\text{avo}} \land \exists\ \{\text{u}_\text{j}\} \in U_C^{u_i} \mid \text{f}_{\text{cd}}(\text{u}_\text{i}, \text{u}_\text{j}, \text{t})\text{=true.}$ After applying CS in every DroS instance at time t_0+T_{iter}:		
$U_{CO}^{u_1}=\{u_1,u_2,u_3\}$	$U_{CO}^{u_2}=\{u_1,u_2,u_3\}$	$U_{CO}^{u_3}=\{u_1,u_2,u_3\}$
indexOf(u_1)in $U_{CO}^{u_1}$=1	indexOf(u_2)in$U_{CO}^{u_2}$=2	indexOf(u_3)in $U_{CO}^{u_3}$=3
$sp_{u_1}=S_p[1]$=25 m/s	$sp_{u_2}=S_p[2]$=20 m/s	$sp_{u_3}=S_p[3]$=15 m/s

Fig. 8. Example of a three UAV conflict. u_1, u_2 and u_3 will collide at p_C if the CAS is not applied at time t_0.

6.3. Example

Fig. 8 shows a conflict scenario with three DroS equipped UAVs at time t_0, used to clarify DroS avoidance methods. Three DroS-equipped UAVs may collide at point p_C if no conflict avoidance is applied. Every 200 ms (T_{iter} is set to 200 ms), DroS re-evaluates the conflict avoidance method and updates its state. For the example, we consider that the three DroS instances are synchronized, so at any time t the three starts simultaneously computing the avoidance method. After t+T_{iter}, all DroS have also finished simultaneously. DroS applies CS in each instance, as $t_{tc}(u_i, u_j, t_0) > t_{avo}(u_i, u_j, t_0)$, every UAV has a collaborative element (e_c) and d(u_i, u_j)< r_C).

The most significant triplets of the KB_{u_1} at iteration t_0 and $t_0 + T_{iter}$ are detailed hereinafter. KB_{u_2} and KB_{u_3} are analogous; therefore, they are not shown.

```
...
drone:iter-t0    owl:Class    drone:Iteration
drone:fcu-iter-t0    owl:Class    drone:FCUInputData
drone:fcu-iter-t0    drone:hasSpeed    22.0f^^xsd:float
drone:adsb-idu2-iter-t0    owl:Class    drone:ADSBInputData
drone:adsb-idu2-iter-t0    drone:hasSpeed    23.0f^^xsd:float
drone:adsb-idu3-iter-t0    owl:Class    drone:ADSBInputData
drone:adsb-idu3-iter-t0    drone:hasSpeed    24.0f^^xsd:float
drone:conflict-idu2-iter-t0    owl:Class
 drone:CollaborativeConflict
drone:conflict-idu3-iter-t0    owl:Class
 drone:CollaborativeConflict
...
drone:state-iter-to-titer    owl:Class    drone:State
drone:state-iter-to-titer    drone:hasState
 "coordinated"^^xsd:string
drone:maneuver-iter-to-titer    owl:Class
 drone:SpeedManeuver
drone:maneuver-iter-to-titer    drone:hasSpeed
 25.0f^^xsd:float
...
```

At time t_1, where $t_1 > t_0 + t_m$, every UAV has started applying CS, changing theirs speeds using the set of target speeds Sp. The conflict scenario is depicted in Fig. 9. KB_{u_1} stores ADS-B messages containing the remote conflict speed at iteration t_1. Compared to the previous one, UAV's speeds have changed following the coordinated strategy. Therefore, it can be inferred that the remote UASs are equipped with DroS, so the conflicts are instances of *DroSConflict*.

```
...
drone:iter-t1    owl:Class    drone:Iteration
drone:fcu-iter-t1    owl:Class    drone:FCUInputData
drone:fcu-iter-t1    drone:hasSpeed    25.0f^^xsd:float
drone:adsb-idu2-iter-t1    owl:Class    drone:ADSBInputData
drone:adsb-idu2-iter-t1    drone:hasSpeed    20.0f^^xsd:float
drone:adsb-idu3-iter-t1    owl:Class    drone:ADSBInputData
drone:adsb-idu3-iter-t1    drone:hasSpeed    15.0f^^xsd:float
...
drone:state-iter-t1    owl:Class    drone:State
drone:state-iter-t1    drone:hasState
 "coordinated"^^xsd:string
drone:maneuver-iter-t1    owl:Class    drone:SpeedManeuver
drone:maneuver-iter-t1    drone:hasSpeed    25.0f^^xsd:float
...
drone:conflict-iter-t1    owl:Class    drone:DroSConflict
drone:conflict-iter-t1    owl:Class    drone:DroSConflict
...
```

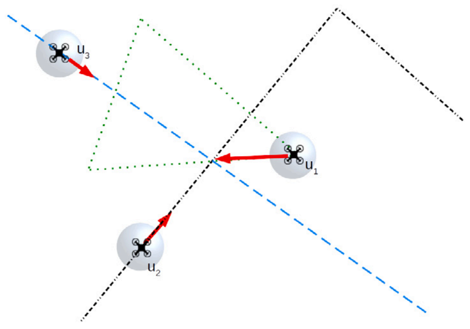
At time t_2, where $t_2 > t_1$, u_1 does not suppose a conflict anymore and recovers its initial speed. However, u_2 and u_3 are still in conflict, as depicted in Fig. 10. As $t_{tc}(u_i, u_j, t_2) < t_{avo}(u_i, u_j, t_2)$, DroS instances of u_2 and u_3 change from CS to EA avoidance method.

At t_2, u_1 no longer is in conflict, so KB_{u_2} is shown hereinafter. KB_{u_2} stores the conflict caused by u_3. As $t_{avo} > t_{tc}$ at iteration $t_2 + T_{iter}$, KB_{u_2} stores an evasive state and an evasive maneuver.

```
...
drone:iter-t2    owl:Class    drone:Iteration
drone:conflict-idu3-iter-t2    owl:Class    drone:DroSConflict
drone:conflict-idu3-iter-t2    drone:hasTavo    10.0f^^xsd:float
drone:conflict-idu3-iter-t2    drone:hasTtc    9.0f^^xsd:float
...
drone:state-iter-t2-Titer    owl:Class    drone:State
drone:state-iter-t2-Titer    drone:hasState
 "evasive"^^xsd:string
...
drone:maneuver-iter-t2-Titer    owl:Class
 drone:EvasiveManeuver
drone:maneuver-iter-t2-Titer    drone:hasBearing
 110.0f^^xsd:float
...
```

EA changes the bearing of u_2 and u_3, represented as vectors $\vec{u}_2$ and $\vec{u}_3$, is depicted in Fig. 11. When conflicts disappear applying EA, DroS returns to the *Planned* state and the FCU recovers the bearing to reach the next waypoint of the initial flight plan, $FP_{u_i}^k$, as depicted in Fig. 12, at time $t_3 > t_2 + T_{iter}$.

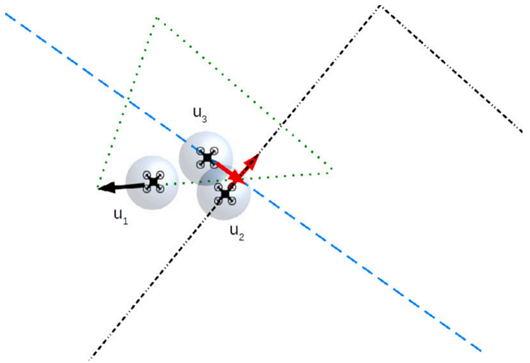
At t_3, KB_{u_2} only stores DroS state (planned) as conflicts were avoided:



n_C=3 at time t_1		
u_1	u_2	u_3
sp_{u_1}=25 m/s	sp_{u_2}=20 m/s	sp_{u_3}=15 m/s
b_{u_1}=260°	b_{u_2}=45°	b_{u_3}=135°

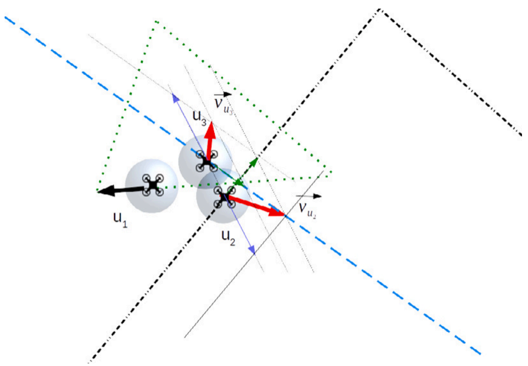
Fig. 9. The three UAV conflict applying the CS strategy at time t_1.

n_C=2 at time t_2



u_1	u_2	u_3
sp_{u_1}=22 m/s	sp_{u_2}=20 m/s	sp_{u_3}=15 m/s
b_{u_1}=260°	b_{u_2}=45°	b_{u_3}=135°
$U_C^{u_1}=\{\emptyset\}$	$U_C^{u_2}=\{u_3\}$	$U_C^{u_3}=\{u_2\}$
$\lvert U_C^{u_1}\rvert = 0$	$\lvert U_C^{u_2}\rvert = 1$	$\lvert U_C^{u_3}\rvert = 1$
	$f_cd(u_2,u_3,t)$=true	$f_cd(u_3,u_2,t)$=true
	$t_{tc}(u_2,u_3)$= 9s	$t_{tc}(u_3,u_2)$= 9s
	$t_{avo}(u_2,u_3)$=10s	$t_{avo}(u_3,u_2)$=10s
	$\mid U_C^{u_i} \mid > 0 \wedge \exists \{u_k\} \in U_C^{u_i} \mid$	
	$t_{tc}(u_i,u_k) \leqslant t_{avo} \wedge f_{cd}(u_i, u_k, t)$=true	
	After applying EA at time $t_2 + T_{iter}$:	
	sp_{u_2}=23 m/s	sp_{u_3}=24 m/s
	b_{u_2}=110°	b_{u_3}=0°

Fig. 10. Two UAVs in conflict applying EA at time t_2.



n_C=2 at time t_2 + T_{iter}		
u_1	u_2	u_3
sp_{u_1}=22 m/s	sp_{u_2}=23 m/s	sp_{u_3}=24 m/s
b_{u_1}=260°	b_{u_2}=110°	b_{u_3}=0°

Fig. 11. Two UAVs in conflict applying EA at time t_2+ T_{iter}.

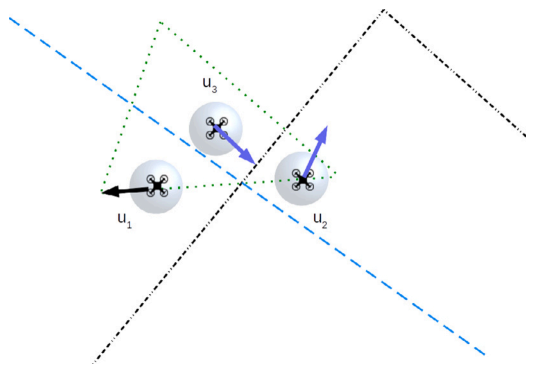
```
...
drone:iter-t3    owl:Class    drone:Iteration
drone:fcu-iter-t3    owl:Class    drone:FCUInputData
drone:fcu-iter-t3    drone:hasBearing    35.0f^^xsd:float
....
drone:state-iter-t3    owl:Class    drone:State
drone:state-iter-t3    drone:hasState    "planned"^^xsd:string
...
```

7. DroS's system simulation

Once implemented DroS, we need to verify that DroS minimizes UAV collisions. Each UAV has its instance of DroS running on an onboard micro-PC, which is part of the UAV payload. The required scenarios to verify DroS are complex to deploy, and UAVs may collide, so we use a simulator called *SIMUdrone*, introduced in Martín-Lammerding et al. (2022a), to verify the collision avoidance.

For such purpose, we perform Hardware In the Loop (HIL) simulations to obtain the time required (Response Time, RT) by the hardware (onboard micro-PC and FCU) to infer the action to be performed according to the scenario and circumstances of the set of UAVs. *SIMUdrone* provides the inputs to be used by the hardware, and the hardware provides the outputs inferred to *SIMUdrone*.

After verifying that the hardware can provide the appropriate responses in a timely and accurate manner, we analyze, with the aid of



n_C=0 at time t_3		
u_1	u_2	u_3
sp_{u_1}=22 m/s	sp_{u_2}=23 m/s	sp_{u_3}=24 m/s
b_{u_1}=260°	b_{u_2}=35°	b_{u_3}=125°

Fig. 12. Conflicts avoided at time t_3.

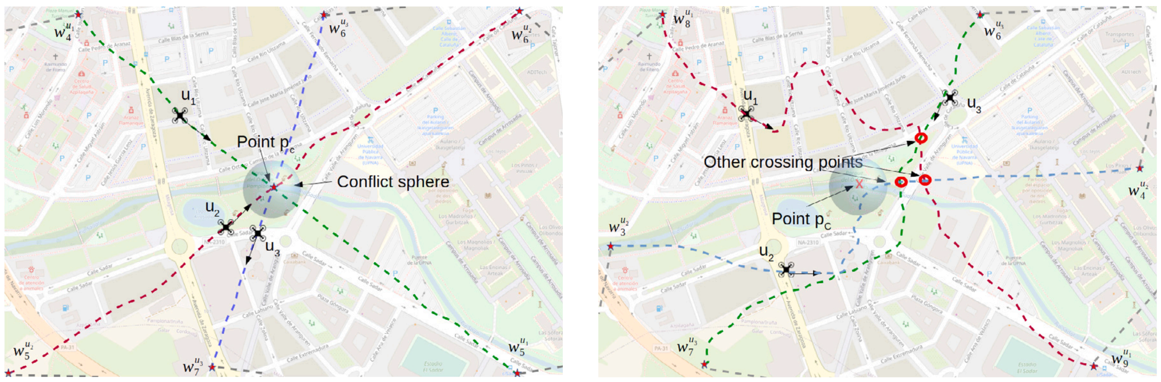


Fig. 13. Conflicting traffic scenarios for three UAVs with rectilinear paths (left) and three UAVs with curved paths (right). Paths in color represent the paths simulated (from $w_j^{u_i}$ to $w_{j+1}^{u_i}$).

the simulator, light to dense conflicting traffic scenarios with and without DroS-equipped UAVs. We consider that a conflicting traffic scenario requires multiple flying UAVs whose flight plans share a common cross point, p_c (point of collision), or at least share a conflict volume around p_c. These scenarios are typical of the VLL aerospace involving T_{copter} UAVs with MTOM values less than 50 kg. The previous waypoint before reaching p_c is denoted as $w_j^{u_i}$, and $w_{j+1}^{u_i}$ is the subsequent waypoint after p_c. The resulting conflicting sphere is centered at p_c with radius r_p. Simulation starts with every UAV located at its waypoint $w_j^{u_i}$ and finishes when every UAV reaches its waypoint $w_{j+1}^{u_i}$ or even collides. Two examples of conflicting traffic scenarios are illustrated in Fig. 13.

7.1. Configuration parameters

As described in Bulusu, Sengupta, Polishchuk, and Sedov (2017), nine is a reasonable number of conflicting UAVs obtained for a dense traffic sphere (100 m radius). Therefore, we simulate traffic scenarios with several conflicting UAVs, $n_C \in [2, 9]$.

The altitude and the nominal speed range are chosen according to the previously indicated scenario, h=50 m (buildings of up to 15 floors), and mean speeds within the range $[15, 25]$ m/s. With regard to the protection, safety and collision spheres, $r_p = 100$ m, $r_s = 50$ m and $r_{cc} = 5$ m are the values used. The GNSS receptors carried by UAVs have a usual location error of ± 5 m, as described in Wing, Eklund, and Kellogg (2005), while compasses may have a deviation of $\pm 3°$, as described in Ladetto and Merminod (2002). Simulations performed consist of 100 different scenarios for all $n_C \in [2, 9]$ conflicting UAVs. The p_c is located in the middle of each scenario. For each scenario, we obtain the $w_j^{u_i}$ and $w_{j+1}^{u_i}$ of each drone u_i. To do this, the scenario is divided into two equal parts whose axis of symmetry passes through p_c. $w_j^{u_i}$ is placed in one of these half-spaces following a uniform probability distribution, and its corresponding $w_{j+1}^{u_i}$ is placed in the opposite direction of the straight line joining $w_j^{u_i}$ with p_c. The distance between $w_j^{u_i}$ and $w_{j+1}^{u_i}$ is obtained by means of another uniform probability distribution in the range $[450, 650]$ m, which is an adequate value for an $r_p = 100$ m. When a $w_j^{u_i}$ is obtained too close to a previous one (when the angles of their relative positions concerning p_c are less than 5°), a new random position is generated.

SIMUdrone allows the use of *steady*, *non-steady*, *DroS*, *DroS-CS* (DroS with only CS avoidance method) and *DroS-EA* (DroS with only EA avoidance method) UAVs. A steady-UAV follows a near rectilinear flight between waypoints ($w_j^{u_i}$ and $w_{j+1}^{u_i}$), with a bearing fluctuation $b_c \pm 6°$. A *non-steady* UAV changes its bearing before arriving p_c, in a given time, t_u, obtained from an uniformly random distributed within 10 s and 15 s, and returns to its original path after a given time, $t_{nu} \in [10, 12]$ s. The turn rate, b_{nc}, is within the range of $\pm 6°/s$ and the speed during the maneuver, sp_{nc}, is within zero and 15 m/s. *Non-steady* UAVs parameters are obtained from realistic and smooth maneuvers. DroS, DroS-CS and DroS-EA UAVs follow a straight path (as *steady* UAV does). Table 7 summarizes the simulation parameters described above.

The target speed set Sp, is defined using speeds within the range sp_{max} and sp_{min}, that are reasonable speeds for a flight type T_{copter}. The speed coordination method implemented in CS resolves a conflict involving multiple UAVs by assigning the highest speed to the UAV with the highest priority ID (ID_{u_i}) while reducing the speeds of the rest of the UAVs. As soon as the related UAV leaves the conflict sphere, CS reassigns the highest speed to the UAV with the remaining highest ID. So that, eventually, all UAVs will manage to pass through the sphere without any collision. Sp is the ordered set of speeds, where the first position corresponds to the highest speed, and the rest are close to the minimum speed (sp_{min}) as shown in Table 8.

Although *SIMUdrone* allows the emulation of different equipment, in this case, the vision camera used is a *Casia* model from the manufacturer Iris Automation. The camera is stabilized so that it points in

Table 7
Main *SIMUdrone* parameters configured.

Parameter	Value	Parameter	Value	Parameter	Value
n_C	[2,9]	Nominal speed range (sp)	[15, 25] m/s	sp_{nc}	[0, 15] m/s
MTOM	< 50 kg	Altitude (h)	50 m	b_{nc}	[6, 12]°/s
Protection radius (r_p)	100 m	Safety radius (r_s)	50 m	t_u	[10, 15] s
Collision radius (r_{cc})	5 m	Distance range from point p_c to $w_j^{u_i}$ and $w_{j+1}^{u_i}$	[450, 650] m	t_{nu}	[10, 12] s
GNSS error	± 5 m			b_c	±3°

Table 8
Simulated DroS parameters for CS avoidance method.

Parameter	Value
sp_{max}	39 m/s
sp_{min}	0.6 m/s
Sp={39, 0.85, 0.85, 0.85, 0.85, 0.6, 0.6, 0.6, 0.6}	

Table 9
Main equipment parameters configured for the simulations.

Parameter	Value
Detection distance (d_{DT})	[1000, 1300] m
Coverage angle, Horizontal–Vertical (θ)	80° – 50°
T_s	50 ms (20 Hz)
Sensibility (1090 MHz)	−61 dBm
Coverage distance (d_{CB})	[1500, 2500] m
T_c	1 s (1 Hz)

the forward direction of the aircraft with an aperture of 80° × 50° and a maximum detection distance of 1300 m. As the detection distance may vary due to atmospheric conditions, its value is obtained from a uniform random distribution between 1000 m and 1300 m.

The low-power ADS-B transceiver used is the model *ping20Si* from the manufacturer uAvionix, which has a transmit power of 20 W. ADS-B standard defines an operation frequency of 1090 MHz and a duty cycle of 1 Hz. Therefore, we calculate the coverage distance, which is 2300 m considering its sensibility of −61 dBm. Fluctuations of received power are modeled by varying the coverage distance using a uniform random distribution within 2000 m and 2300 m. *Steady* and *non-steady* UAVs do not require any detecting sensor or collaborative element. DroS-CS requires an ADS-B transceiver, while DroS-EA requires a vision camera, and DroS requires both. Table 9 summarizes these configuration parameters.

DroS duty cycle is adjusted to the lowest period of the available data sources (ADS-B or vision camera). Therefore, T_{iter} is set to 50 ms. DroS response time depends on its state (since the number of SPARQL executed varies for each state) and on the size of the KB (defined by the number of triplets stored in each iteration and the number of iterations that are kept for inference purposes, as described in Martín-Lammerding et al., 2021). The number of iterations stored in the KB (n_{iter}) is determined by the UAV, which traverses the conflict sphere with the highest speed. In this case, n_{iter} is set to 100 since the UAV traverses 200 m at a speed of 40 m/s.

The Key Performance Indicators (KPIs) used to assess collision avoidance are the UAV Conflict Avoidance rate (UCA%), defined as the number of scenarios without any collision divided by the number of total scenarios simulated. DroS Conflict Avoidance rate (DCA%), defined as the number of scenarios without any DroS-equipped UAV collided divided by the number of total scenarios simulated.

7.2. HIL simulation

HIL simulation is performed using a Pi3 (Raspberry Pi Foundation, 2022) companion computer and a Pixhawk PX4 FCU (PX4, 2022) as shown in Fig. 14. Different UAV equipment combinations are proposed to obtain $DroS_{u_i}$'s RT at each state. While the *Planned* state requires

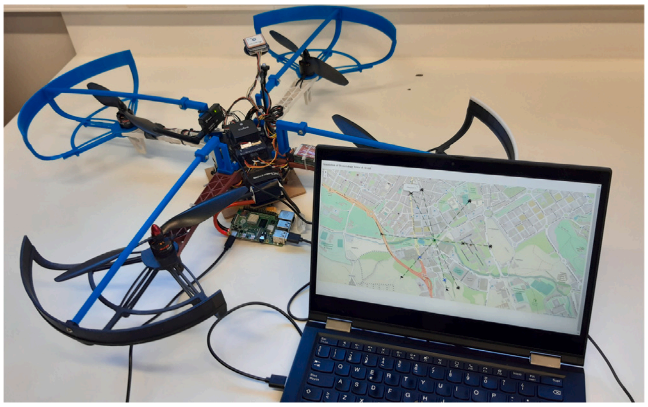


Fig. 14. HIL simulation framework.

Table 10
DroS RT (ms) for each state, measured in a Pi3.

n_C	DroS states RT					
	Planned		Coordinated		Evasive	
	Mean	Sdev	Mean	Sdev	Mean	Sdev
2	8.26	1.41	156.56	2.40	201.00	3.57
3	8.25	1.35	158.73	3.07	201.20	3.16
4	8.36	1.56	158.21	2.82	200.73	2.95
5	**8.24**	1.34	158.48	2.83	200.97	2.97
6	8.30	1.59	158.30	2.89	200.79	2.99
7	8.27	1.38	159.15	3.32	201.83	3.43
8	8.66	1.61	158.87	2.94	201.70	2.53
9	8.70	1.90	159.41	2.61	**201.88**	3.60

the absence of conflicts, $n_C - 1$ conflicting UAVs are required for both the *Coordinated* (DroS-equipped UAVs) and the *Evasive* (*steady* UAVs) states. Table 10 depicts the mean and the standard deviation values of RT for each state, obtained from 100 simulated scenarios, after discarding some initial scenarios to warm up. After the HIL simulation, RT values of no more than 202 ms are obtained. The lowest mean value observed corresponds to the *Planned* state (8.24 ms), as it performs a single SPARQL query. In comparison, the highest mean value corresponds to the *Evasive* state (201.88 ms) since it requires executing multiple geoprocessing operations and SPARQL queries. The stability of RT values is also observed regardless of the number of conflicts considered. Therefore DroS reaction time does not degrade as the number of simultaneous conflicts increases.

Finally, we re-configure DroS according to the RT values obtained, increasing T_{iter} to 202 ms. The choice of a shorter period would have implied a greater number of iterations that hardly provide relevant information to the system, as will be discussed below.

SIMUdrone follows three Gaussian distributions, one for each of the three possible states (*Planned*, *Coordinated* and *Evasive*). For each state, the mean and standard deviation considered correspond to the worst case observed (with the higher deviation) as depicted in Table 11.

7.3. Conflict scenarios simulation

In this section, we analyze the performance of DroS with the configuration previously described. First, we assume that all the conflicting

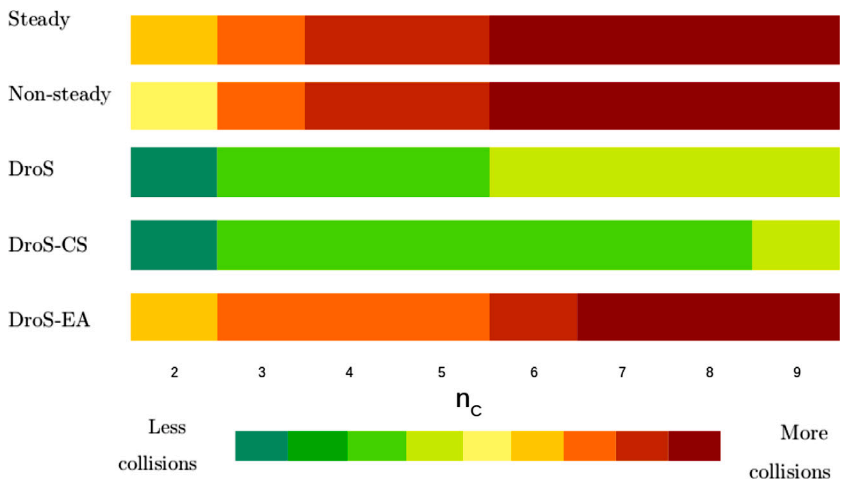


Fig. 15. Collisions heat-map for comparing homogeneous simulations results.

Table 11
Gaussian distribution parameters used to obtain the RT for Coordinated, Evasive and Planned states.

DroS state	Gaussian distribution	
	Mean	Sdev
Planned	8.70	1.90
Coordinated	159.15	3.32
Evasive	201.88	3.60

UAVs involved have the same equipment, that is, all UAVs are *steady*, *non-steady*, DroS, DroS-CS, or DroS-EA. After that, we focus on scenarios where the conflicting UAVs have different equipment, which is the most likely and realistic scenario.

7.3.1. Homogeneously equipped UAVs

Table 12 shows the collision rate metrics for 100 simulated scenarios for each UAV type. While Fig. 15 depicts the collision results obtained as a heatmap to easily compare the five UAV types.

The number of collided UAVs tends to n_C for *steady* UAVs, as all trajectories converge at point p_c. However, some simulations are collision-free, although *steady* UAVs do not carry a CAS. This occurs when less than four UAVs are in conflict and the distribution of speeds, bearing, and distances to the point p_c are favorable. When $n_C \geq 4$ is less likely to avoid a collision by favorable parameters randomness.

Non-steady UAVs do not converge at point p_c and this may avoid collisions. However, their random bearings cause other crossings points at different locations that may cause collisions. When the number of UAVs involved is low, there is a lower probability of collision, and simulations show similar values to *steady* UAVs. As the number of UAVs grows, $n_C \geq 4$, the number of scenarios with collisions increases, but the values obtained do not differ too much from those of *steady* UAVs.

DroS equipped UAVs suffer fewer collisions than *steady* and *non-steady* UAVs do. According to the type of avoidance, DroS-CS has the highest DCA rate, while DroS-EA has the lowest one. DroS-EA changes the heading of the UAVs and with them, also changes the pointing of the vision camera so that a previously detected conflict may disappear or a new, previously undetected conflict may appear. This means that DroS-EA may lose awareness due to bearing changes, which may cause an increased risk of collision. In addition, fluctuation in conflict detection results in multiple path changes that cause trajectory clutter, which also increases the risk of collisions. Note that flight paths are obtained considering conflicts as electrostatic charges that repel the UAV, so a change in the number of conflicts means a change in the path.

DroS-CS avoidance method resolves a multiple UAV conflict by selecting the UAV with the highest priority, which will traverse the conflict sphere. In contrast, other UAVs reduce their speed to allow the first UAV to leave the conflict sphere. This process is repeated until all UAVs leave the conflicting sphere. Therefore, DroS-CS collisions are due to a lack of time to prioritize all conflicting UAVs, as multiple remaining conflicting UAVs may reach p_c and collide.

DroS collisions are mainly caused due to remaining conflicting UAVs, those not coordinated already by CS but still in conflict, that are close to p_c, and, therefore close to colliding. In this situation, DroS apply EA to change the bearings as a last resort to avoid collisions. As seen in Table 12, the EA method is not always able to avoid collisions when the number of UAVs involved is greater or equal to four. EA may not find a free-collision path for every UAV to traverse the conflict sphere. Instead, EA may cause a mess of paths, increasing the collision risk. However, DroS applies EA but improves DroS-EA performance as it performs a fusion of conflict sources (ADS-B + vision camera) that provides complete situational awareness and overcomes the vision camera limitation.

Thus all DroS-equipped UAVs minimize collisions significantly. As shown in Table 12, when $n_C \geq 6$ UAVs without DroS have a collision probability greater than 75%. With DroS-equipped UAVs, the probability is reduced below 30%. Predictably, having collaborative equipment on all aircraft gives us a great advantage in avoiding collisions, but it may not be a realistic scenario. Therefore, what happens when not all UAVs involved in conflicts have the appropriate collaborative equipment is discussed below.

7.3.2. Heterogeneously equipped UAVs

Table 13 summarizes the results obtained for 100 simulated scenarios with multiple equipment combinations. Where three different types of UAVs are considered: *steady*, *non-steady* and DroS-equipped. Note that each case study is denoted by $\langle xyz \rangle$, being x the number of *steady* UAVS, y the number of *non-steady* UAVs and z the number of DroS-equipped UAVs. In order to better understand this table, Fig. 16 depicts an example of an scenario ($\langle 111 \rangle$) with three UAVs where u_1 is a *steady* UAV, u_2 is a *non-steady* UAV and u_3 is a DroS-equipped UAV that applies the avoidance method EA. Paths in color are the simulated ones (path from $w_j^{u_i}$ to $w_{j+1}^{u_i}$) and those in gray are not.

In these scenarios, DroS UAVs share airspace with non-collaborative UAVs (*steady* or *non-steady* UAVs) that only can be detected with the vision camera and avoided by the EA method. As expected, the presence of non-collaborative UAVs implies a reduction in collision avoidance rates. Collisions are significantly minimized when DroS-equipped UAVs predominate since DroS UAVs can apply conflict data fusion and collaborative avoidance with other DroS UAVs. As previously observed,

Table 12
Homogeneously equipped UAVs: simulation results.

n_C	Steady			Non-steady			DroS			DroS-CS			DroS-EA		
	UCA%	UAV collided		UCA%	UAV collided		DCA%	UAV collided		DCA%	UAV collided		DCA%	UAV collided	
		Mean	Sdev		Mean	Sdev		Mean	Sdev		Mean	Sdev		Mean	Sdev
2	59	2.00	0.00	62	2.00	0.00	99	2.00	0.00	100	0.00	0.00	64	2.00	0.00
3	13	2.01	0.11	11	2.00	0.00	78	2.00	0.00	97	2.00	0.00	3	2.01	0.10
4	0	2.57	0.89	0	2.76	0.95	69	2.06	0.36	90	2.00	0.00	0	2.70	0.95
5	0	3.68	0.72	0	3.65	0.78	64	2.17	0.56	95	2.00	0.00	0	3.74	0.72
6	0	4.80	0.97	0	4.69	1.15	38	2.24	0.99	70	3.29	1.02	0	4.75	0.99
7	0	5.70	0.76	0	5.77	0.65	32	3.00	1.22	88	2.50	0.90	0	5.51	0.86
8	0	6.81	1.13	0	6.90	1.01	25	2.89	1.21	84	2.25	0.68	0	6.94	1.06
9	0	7.75	0.82	0	7.58	0.90	20	3.13	1.30	66	2.29	0.72	0	7.71	0.76

Table 13
Heterogeneously equipped UAVs: simulation results.

n_C	Case	DCA%	UCA%	# collided		n_C	Case	DCA%	UCA%	# collided		n_C	Case	DCA%	UCA%	# collided	
				Mean	Sdev					Mean	Sdev					Mean	Sdev
2	⟨101⟩	90	90	2.00	0.00	6	⟨141⟩	89	0	3.58	1.08	8	⟨026⟩	14	9	3.47	1.38
2	⟨011⟩	89	89	2.00	0.00	6	⟨204⟩	61	49	2.63	1.02	8	⟨035⟩	46	19	2.89	1.14
3	⟨012⟩	97	97	2.00	0.00	6	⟨213⟩	71	25	2.47	0.91	8	⟨044⟩	45	6	3.59	1.42
3	⟨021⟩	89	59	2.00	0.00	6	⟨222⟩	77	8	2.90	1.11	8	⟨053⟩	62	2	3.87	1.43
3	⟨102⟩	97	97	2.00	0.00	6	⟨231⟩	90	0	3.74	0.92	8	⟨062⟩	69	0	4.89	1.43
3	⟨111⟩	90	57	2.00	0.00	6	⟨303⟩	73	21	2.42	0.81	8	⟨071⟩	95	0	5.76	0.92
3	⟨201⟩	84	58	2.00	0.00	6	⟨312⟩	78	11	2.74	1.07	8	⟨107⟩	35	35	2.74	1.09
4	⟨013⟩	87	87	2.15	0.55	6	⟨321⟩	92	0	3.6	0.95	8	⟨116⟩	16	9	3.23	1.45
4	⟨022⟩	88	66	2.06	0.34	6	⟨402⟩	88	5	2.63	0.91	8	⟨125⟩	56	16	3.00	1.26
4	⟨031⟩	90	18	2.12	0.46	6	⟨411⟩	85	0	3.79	1.03	8	⟨134⟩	52	7	3.25	1.28
4	⟨103⟩	89	89	2.00	0.00	6	⟨501⟩	88	0	3.64	1.12	8	⟨143⟩	46	1	4.11	1.43
4	⟨112⟩	96	75	2.00	0.00	7	⟨016⟩	9	9	2.91	1.11	8	⟨152⟩	73	0	4.67	1.23
4	⟨121⟩	82	22	2.24	0.65	7	⟨025⟩	57	39	2.43	0.83	8	⟨161⟩	95	0	5.63	0.85
4	⟨202⟩	87	68	2.25	0.67	7	⟨034⟩	64	37	2.79	1.05	8	⟨206⟩	11	6	3.32	1.51
4	⟨211⟩	87	17	2.12	0.45	7	⟨043⟩	61	5	3.21	1.33	8	⟨215⟩	53	19	2.84	1.30
4	⟨301⟩	86	18	2.30	0.70	7	⟨052⟩	72	1	3.74	1.07	8	⟨224⟩	50	9	3.46	1.54
5	⟨014⟩	55	55	2.00	0.00	7	⟨061⟩	92	0	4.67	1.04	8	⟨233⟩	59	0	3.80	1.29
5	⟨023⟩	73	59	2.34	0.76	7	⟨106⟩	14	14	2.81	1.08	8	⟨242⟩	77	0	4.86	1.26
5	⟨032⟩	82	30	2.17	0.56	7	⟨115⟩	52	38	2.45	0.92	8	⟨251⟩	91	0	5.66	1.00
5	⟨041⟩	91	4	2.74	0.99	7	⟨124⟩	51	13	2.76	1.19	8	⟨305⟩	53	21	2.68	1.24
5	⟨104⟩	58	58	2.10	0.43	7	⟨133⟩	68	8	2.85	1.23	8	⟨314⟩	59	5	3.32	1.45
5	⟨113⟩	82	67	2.36	0.78	7	⟨142⟩	75	1	3.75	1.22	8	⟨323⟩	59	0	3.87	1.45
5	⟨122⟩	84	27	2.22	0.63	7	⟨151⟩	88	0	4.83	1.05	8	⟨332⟩	77	0	4.85	1.44
5	⟨131⟩	92	2	2.64	0.93	7	⟨205⟩	47	34	2.67	1.07	8	⟨341⟩	93	0	5.48	1.07
5	⟨203⟩	75	51	2.29	0.71	7	⟨214⟩	61	22	2.74	1.21	8	⟨404⟩	49	7	3.46	1.45
5	⟨212⟩	81	30	2.27	0.72	7	⟨223⟩	65	5	2.81	1.02	8	⟨413⟩	59	2	3.81	1.37
5	⟨221⟩	90	2	2.91	0.98	7	⟨232⟩	77	1	3.80	1.06	8	⟨422⟩	83	0	4.59	1.11
5	⟨302⟩	80	30	2.33	0.74	7	⟨241⟩	86	0	4.67	1.11	8	⟨431⟩	88	0	5.78	0.92
5	⟨311⟩	89	4	2.92	0.99	7	⟨304⟩	51	13	2.80	1.24	8	⟨503⟩	57	1	3.58	1.25
5	⟨401⟩	88	3	2.78	0.98	7	⟨313⟩	71	6	2.93	1.23	8	⟨512⟩	76	0	4.87	1.32
6	⟨015⟩	43	43	2.32	0.74	7	⟨322⟩	75	1	3.99	1.26	8	⟨521⟩	95	0	5.55	1.02
6	⟨024⟩	59	43	2.46	0.85	7	⟨331⟩	94	0	4.53	1.12	8	⟨602⟩	79	0	4.50	1.34
6	⟨033⟩	78	35	2.51	0.90	7	⟨403⟩	62	9	3.04	1.24	8	⟨611⟩	89	0	5.80	0.96
6	⟨042⟩	76	8	2.97	1.12	7	⟨412⟩	79	0	3.56	1.26	8	⟨701⟩	92	0	5.72	1.03
6	⟨051⟩	91	0	3.61	1.02	7	⟨421⟩	92	0	4.73	1.21	9	⟨018⟩	20	20	3.03	1.15
6	⟨105⟩	61	61	2.62	0.94	7	⟨502⟩	75	0	3.51	1.26	9	⟨027⟩	21	13	2.97	1.09
6	⟨114⟩	70	48	2.38	0.80	7	⟨511⟩	83	0	4.79	1.20	9	⟨036⟩	9	6	4.02	1.66
6	⟨123⟩	69	28	2.60	1.03	7	⟨601⟩	93	0	4.56	0.99	9	⟨045⟩	41	8	3.68	1.50
6	⟨132⟩	76	5	2.88	1.11	8	⟨017⟩	35	35	2.89	1.12	9	⟨054⟩	61	4	3.97	150
9	⟨063⟩	63	0	4.9	1.23	9	⟨225⟩	32	4	3.54	1.31	9	⟨711⟩	94	0	6.60	1.08
9	⟨072⟩	73	0	5.8	1.40	9	⟨234⟩	46	0	4.14	1.51	9	⟨252⟩	71	0	5.88	1.33
9	⟨081⟩	93	0	7.01	1.15	9	⟨243⟩	67	1	4.76	1.36	9	⟨306⟩	11	3	3.97	1.42
9	⟨108⟩	16	16	3.14	1.33	9	⟨261⟩	92	0	6.54	1.12	9	⟨324⟩	47	0	4.31	1.64
9	⟨117⟩	32	27	3.07	1.26	9	⟨315⟩	50	4	3.34	1.46	9	⟨342⟩	64	0	5.96	1.20
9	⟨126⟩	6	4	4.28	1.66	9	⟨333⟩	61	0	4.70	1.28	9	⟨405⟩	44	3	3.65	1.63
9	⟨135⟩	36	6	3.65	1.47	9	⟨351⟩	89	0	6.83	1.09	9	⟨423⟩	56	0	4.85	1.45
9	⟨144⟩	53	0	4.02	1.48	9	⟨414⟩	46	1	4.32	1.56	9	⟨441⟩	88	0	6.98	1.09
9	⟨153⟩	56	0	4.76	1.22	9	⟨432⟩	74	0	5.74	1.18	9	⟨513⟩	60	0	4.66	1.60
9	⟨162⟩	76	0	5.65	1.20	9	⟨504⟩	49	1	4.21	1.52	9	⟨531⟩	93	0	6.73	1.07
9	⟨171⟩	92	0	6.78	1.11	9	⟨522⟩	78	0	5.76	1.11	9	⟨612⟩	68	0	5.88	0.99
9	⟨207⟩	27	17	3.34	1.40	9	⟨603⟩	56	0	4.76	1.58	9	⟨702⟩	80	0	5.87	1.22
9	⟨216⟩	4	3	4.23	1.56	9	⟨621⟩	91	0	6.78	1.07	9	⟨801⟩	94	0	6.87	1.03

steady and *non-steady* UAVs may avoid collisions when considering a low number of conflicting UAVs ($n_C < 4$), mainly to the randomness of speed, of the distance to p_c or/and of the heading fluctuations.

The difference between the DCA and UCA KPIs represents DroS collision avoidance performance in a given scenario. The most significant differences occur as a greater proportion of non-DroS UAVs are

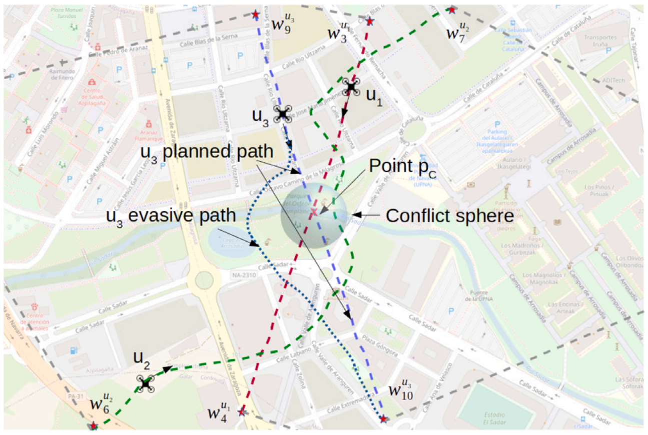


Fig. 16. Planned and evasive avoidance of three UAVs.

observed. DroS UAVs are very likely to avoid a collision, while non-DroS UAVs are more likely to collide. The maximum difference (case ⟨521⟩ or ⟨071⟩) occurs when the scenario includes just one DroS that manages to avoid colliding.

Fig. 17 includes two heatmaps depicting the collision results obtained. Heatmap (1) represents collisions between *steady* UAVs and DroS-equipped UAVs, while Heatmap 2 depicts collisions between *non-steady* UAVs and DroS-equipped UAVs. As can be observed, when DroS UAVs predominate in a given scenario, the number of collisions is reduced, so the heatmap turns greener near the X axis. As an example of the above, the simulation of a scenario with four DroS-equipped UAVs and one *steady* UAV has fewer collisions than the scenario with four *steady* UAVs and one DroS-equipped UAV.

Simulations with DroS-CS and DroS-EA were also performed. Finally, we summarize the results obtained hereinafter. Predictably, DroS-CS's DCA is like DroS's DCA when traffic is mostly DroS-CS UAVs. At the same time, DroS-CS UAVs are outnumbered, *steady* and *non-steady* UAVs conflicts are undetected, decrementing the value of the DCA indicator.

DroS-EA reduces collisions as DroS does but only when $n_C \leq 3$, while, in the rest of the cases, bearing changes and limited conflict detection increase collisions.

7.3.3. Discussion

DroS-equipped UAVs minimize the number of collisions in comparison with any other UAV type, even in dense traffic scenarios where airspace is shared. We increase the complexity of the scenarios (high density and high heterogeneity of UAVs) to force high collision risk situations to verify the avoidance capabilities of the proposed CAS. Since multiple configurations and paths may be considered, we use the worst-case criteria to select configuration parameters, like the maximum n_C or the UAVs paths that cross at p_c.

As we have seen, the combination of collaborative avoidance and conflict data fusion are key factors in minimizing collisions. Moreover, collaborative data plays a decisive role in improving situational awareness to minimize collisions, as was noted in the simulations performed. Therefore, we consider that DroS T_{iter} can be adjusted to the duty cycle of the collaborative element, avoiding unnecessary DroS executions that increase KB size without valuable knowledge.

DroS has the highest DCA rates of the simulations performed because it implements collaborative avoidance and conflict data fusion. However, DroS avoidance capabilities can be further improved using its knowledge to implement and combine the current available with other conflict avoidance methods.

8. Conclusions and future work

In this paper, we have proposed a novel autonomous, distributed, and ontology-driven CAS for UAVs called DroS. It is based on the Dronetology-cas ontology, so it is a knowledge-driven decision system. We have implemented DroS over a RaspBerry Pi3 companion computer, obtaining an adequate response time even with limited computational capabilities.

Two strategies for conflict avoidance have been developed and verified with the aid of *SIMUdrone*, which consists of an air traffic simulator and hardware in the loop framework. Different combinations of collaborative and non-collaborative UAVs are evaluated, and interesting experimental results have been obtained, showing that DroS-equipped UAS can share airspace safely. We have observed that using DroS, configured with both strategies, significantly minimizes collisions, even in dense traffic airspace and complex situations. For example, DroS-equipped UASs reduce their collision probability below 30% even in a dense traffic scenario with up to six UASs. Another example is a traffic scenario with nine heterogeneously-equipped UAS, where a DroS-equipped UAS has only a probability of 5% of colliding. Note that DroS achieves these results without requiring altitude changes since these changes are usually the first option to be used in dense and saturated environments.

The use of ontology allows automatic and machine-interpretable knowledge sharing. DroS's knowledge increases while UAS flies, improving its situational awareness. It infers decisions autonomously as the triplets are a machine-interpretable format. DroS' knowledge can be shared with other DroS instances. DroS reduces the amount of data that is required to interchange between UAS in order to coordinate themselves, as implicit knowledge is inferred in every DroS-equipped UAS. Additionally, DroS is a deterministic system that is foreseeable

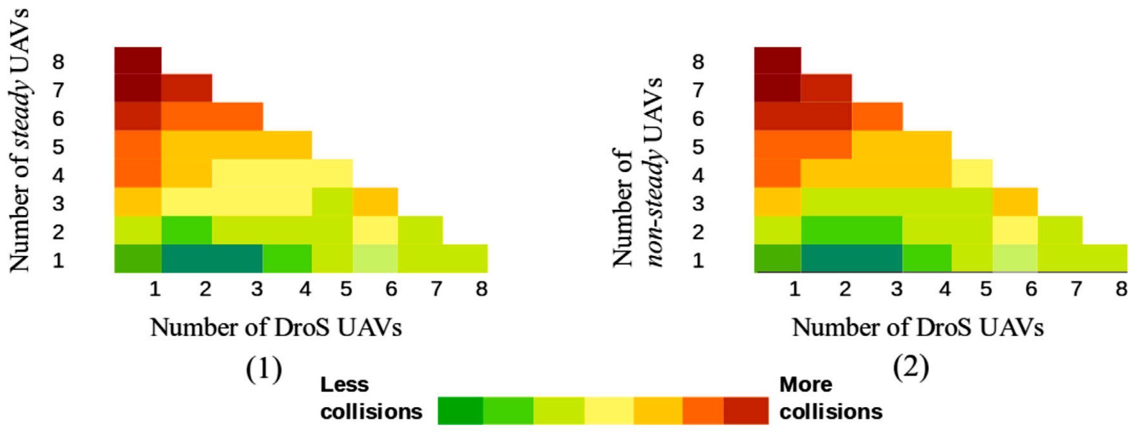


Fig. 17. Collisions heatmap for heterogeneous UAVs simulations.

as its conflict resolution strategies are known, so that other DroS can anticipate maneuvers. This reduces the complexity of the collaborative network protocol requiring only to broadcast messages as ADS-B does. So, decisions are improved, and collisions are avoided or minimized.

In future work, DroS avoidance capabilities can further be improved using the knowledge managed to implement other new strategies and combine all of them. We also plan to consider additional strategies that allow taking into account possible failures (mechanical, electrical, electronic, or communication) in the aircraft that make the current methods not work properly. Maneuvers consisting of altitude changes will be considered for some UAS typologies that fly in higher airspace. The application to other types of vehicles (cars, ships, submarines, rovers, etc.) will also be considered as Dronetology-cas can be adapted easily to consider knowledge required for another vehicle type. Other hardware with more computation capacity, like *Nvidia Jetson*, will be considered to reduce response time, so it will update DroS situational awareness more frequently and improve its decisions. Additionally, in terms of machine learning capabilities, we consider it appropriate to build an annotated repository that collects the knowledge acquired about the flights performed and the conflicts detected with the aim of autonomous training systems.

CRediT authorship contribution statement

David Martín-Lammerding: Investigation, Formal analysis, Experimentation, Data curation, Software development, Writing – original draft, Writing – review & editing. **José Javier Astrain:** Conceptualization, Investigation, Formal analysis, Methodology, Writing – review & editing, Project management. **Alberto Córdoba:** Conceptualization, Investigation, Formal analysis, Methodology, Writing – review & editing, Funding acquisition. **Jesús Villadangos:** Investigation, Verification, Writing – review & editing, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work has been supported in part by the Ministerio de Ciencia e Innovación (Spain) under the research grant RTI2018-095499-B-C31 IoTrain; in part by Agencia Estatal de Investigación (AEI) and European Union NextGenerationEU/PRTR PLEC2021-007997: Holistic power lines predictive maintenance system; and in part by the Government of Navarre (Departamento de Desarrollo Económico) under the research grants 0011-1411-2021-000021 EMERAL: Emergency UAVs for long range operations, 0011-1365-2020-000078 DIVA, and 0011-1411-2021-000025 MOSIC: Plataforma logística de largo alcance, eléctrica *y* conectada.

Appendix

This appendix contains the formal definitions of the elements that make up the DroS core and a table summarizing the symbols used throughout the document.

Definition 1. Let a UAS be a unmanned aircraft, the equipment to control it remotely, including the control link, and the payload. A UAS may be autonomous or human-operated with a control system. The unmanned aircraft main parts are the auto-pilot, the flight control unit (FCU), multiple sensors and the communication equipment. Altimeters and Global Navigation Satellite Systems (GNSS) are the more widely used onboard sensors.

Definition 2. Let Nv be the geometrical plane in an XYZ coordinate system where UASs fly. This plane is parallel to XY, at a given height h, such that $h > 0$.

Definition 3. Let U be the set of all UAS in Nv, $U = \{u_i, i \in \mathbb{N}\}$. Each UAS $\{u_i\} \in U$, has a speed at a given time t, obtained from the function $sp(u_i,t)=v \in \mathbb{R}$, and a location defined by coordinates (x, y, h), obtained from the function $xy(u_i,t) = (x, y) \in \mathbb{R}^2$, $\forall u_i \in U$. Every $\{u_i\} \in U$ has a unique identifier id_{u_i}.

Definition 4. A UAS flies over some particular locations in Nv, called waypoints $\{w_j\}$. A waypoint is a location determined by GNSS. Let W be the set of waypoints, $W=\{w_j, j \in \mathbb{N}\}$, $W \subset Nv$.

Definition 5. Let $FP^k_{u_i}$ be the flight plan of a UAS $\{u_i\}$ with k waypoints, $k \in \mathbb{N}$, $\{u_i\} \in U$, $\forall u_i \in U$ is a totally ordered set (may have duplicate elements) created from W | $FP^k_{u_i} = \{w_1, w_2, \dots, w_k\}$, j:1..k, $j \in \mathbb{N}$. $FP^k_{u_i}$ can also define a set of speeds Sp= $\{sp_1, sp_2, \dots, sp_j\}$, j:1..k-1, $j \in \mathbb{N}$, $\{u_i\}$ applies during the flight between waypoints. The take-off location or HOME is the first waypoint in $FP^k_{u_i}$, $\{w_1\} \in Nv$. The landing location, that usually coincides with HOME, is the last waypoint in $FP^k_{u_i}$, $\{w_k\} \in Nv$.

Definition 6. Let T_P be the set of UAS types according to the flight mode. $T_P= \{ T_{copter}, T_{fixed\text{-}wing}, T_{vtol} \} \mid \{u_i\} \in T_{copter}$ OR $\{u_i\} \in T_{fixed\text{-}wing}$ OR $\{u_i\} \in T_{vtol}$, $\forall u_i \in U$. Let $T^{u_i}_P$ be the T_P of $\{u_i\}$.

Definition 7. DroS(u_i) is an autonomous and distributed CAS, based on the ontology Dronetology-cas and deployed in $\{u_i\} \in U$. $\{DroS_{u_i}\} \in Eq_{u_i}$, $\forall u_i \in U$. DroS runs continuously in an infinite loop with a minimum period of T_{iter}. A DroS's iteration is a complete execution of the loop. Iterations are numbered sequentially starting in 1 until the $FP^k_{u_i}$ is over. DroS's KB is a repository of knowledge about $U^{u_i}_C$.

Definition 8. Let Eq_{u_i} be the set of equipment installed on $\{u_i\} \in U$, $Eq_{u_i}= \{eq_n, n \in \mathbb{N}\}$, $\forall u_i \in U$. The equipment $\{eq_n\}$ can be of two types: collaborative (ce_j) or non-collaborative (s_z).

Definition 9. Let U_{dros} be the subset of DroS-equipped UAS, $U_{dros} \subset U$, $U_{dros} = \{ u_i \mid Eq_{u_i} = \{ce_j, s_z, DroS_{u_i}\}, \forall u_i \in U \}$.

Definition 10. A collision detection function $f_{cd}(u_i, U^{u_i}_C, t)$= {true, false} implemented in $DroS_{u_i}$ is a boolean time-dependent function that assess if a remote $\{u_j\}$ might collide in a given time t, $\forall u_j \in U^{u_i}_C$. As an example, a collision detection function $f_{cd}(u_i, U^{u_i}_C, t)$ could be implemented using the CBDR technique.

Definition 11. Let $d(u_i,u_j,t)$ be the euclidean distance between $\{u_i\}$, $\{u_j\} \in U$, $j \neq i$, at a given time t, $\forall u_i, u_j \in U$.

Definition 12. Let $t_{tc}(u_i, u_j)$ be the worst-case time to collision between $\{u_i\}$ and $\{u_j\}$, where $t_{tc}(u_i, u_j)=d(u_i, u_j,t)/speed_{relative}(u_i, u_j, t)$, $i \neq j$, at a given time t, $\forall u_i, u_j \in U_C$.

Definition 13. Let t_{avo} be the avoidance time before a collision occurs that allows a conflict avoidance, where $t_{avo}(u_i, u_j, t) = \frac{r^{u_i}_S}{speed_{relative}(u_i,u_j,t)}$ at a given time t.

Table 14
Table of symbols and abbreviations.

(x, y, z) or xyz	Coordinates	UAS	Unmanned Aircraft System
$\overline{v}$	Velocity vector	UAV	Unmanned Aircraft Vehicle
s	Speed, velocity modulus	CAS	Collision Avoidance System
id	Unique UAV identificator	ADS-B	Automatic Dependent Surveillance Broadcast
t	Time	KB	Knowledge Base
b	Bearing	RT	Response Time
br	Relative bearing	FCU	Flight Control Unit
u	A UAV	GNSS	Global Navigation Satellite System
T_P	Set of UAVs types	GPS	Global Positioning System
s_z	zth onboard sensor	CS	Coordinated Speed
ce	Collaborative element	EA	Evasive Action
Eq	UAV equipment set	PF	Potential Field
$V_C^{u_i}$	Volume where are located u_i conflicts	CBDR	Constant bearing-decreasing range
V_P	Protection volume	MTOM	Maximum Take Off Mass
T_c	Collaborative element period	VLL	Very Low Level airspace
S_p	Set of target speeds	BVLOS	Beyond Visual Line of Sigh
t_{tc}	Time to collision	VTOL	Vertical Take-off and Landing
t_{avo}	Time of avoidance	STOL	Short Take-off and Landing
r_P	Protection radius	CTOL	Conventional Take-off and Landing
r_S	Security radius		
V_S	Security volume		
r_{CC}	Collision radius		
V_{CC}	Collision volume		
r_D	Sensor detection radius		
V_D	Sensor detection volume		
r_{CB}	Coberture radius		
V_{CB}	Coberture volume		
T_{iter}	Iteration period		
θ	Aperture angle of a camera		
$U_C^{u_i}$	Conflicts of u_i		
n_C	The cardinal of U_C		
$U_{CO}^{u_i}$	Ordered conflicts by id		
U_{ce}	Collaborating UAVs set		
U_{nce}	Non-collaborating UAVs set		
U_{dros}	DroS equipped UAVs set		
f_{cd}	Collision detection function		
pos_{u_i}	Position of u_i in a set		

Definition 14. Let S_p be the set of speeds { $sp_1, sp_2, \ldots, sp_0$ }, o:1..n_C, $o \in \mathbb{N}$, | $\forall\, sp_o \in S_p$, $sp_o \geq sp_{o+1}$.

Table 14 shows on the left column a list of symbols with their meaning used in this paper and, in the right column, a list of abbreviations.

References

Alejo, D., Cobano, J., Heredia, G., & Ollero, A. (2014). Optimal reciprocal collision avoidance with mobile and static obstacles for multi-UAV systems. In *2014 International conference on unmanned aircraft systems* (pp. 1259–1266). IEEE.

Apache Foundation (2022a). Apache Jena. https://jena.apache.org/. (Accessed 12 May 2022).

Apache Foundation (2022b). ARQ Jena. https://jena.apache.org/documentation/query/library-function.html. (Accessed 12 May 2022).

Barrado, C., Boyero, M., Brucculeri, L., Ferrara, G., Hately, A., Hullah, P., et al. (2020). U-Space concept of operations: A key enabler for opening airspace to emerging low-altitude operations. *Aerospace*, *7*(3), 24.

Barredo-Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115.

Beckett, D., & Berners-Lee, T. (2008). Turtle - terse RDF triple language, W3C team submission. See: http://www.w3.org/TeamSubmission/turtle/.

Brooker, P. (2013). Introducing unmanned aircraft systems into a high reliability ATC system. *The Journal of Navigation*, *66*(5), 719–735.

Bugallo, M. F., Xu, S., & Djurić, P. M. (2007). Performance comparison of EKF and particle filtering methods for maneuvering targets. *Digital Signal Processing*, *17*(4), 774–786.

Bulusu, V., Sengupta, R., Polishchuk, V., & Sedov, L. (2017). Cooperative and non-cooperative UAS traffic volumes. In *2017 International conference on unmanned aircraft systems* (pp. 1673–1681). IEEE.

Capitán, J., Merino, L., & Ollero, A. (2011). Coordination of multiple UAS for tracking under uncertainty. In *Proceedings of the 1st workshop on research, education and development on unmanned aerial systems* (p. Poster Session).

Chand, B. N., Mahalakshmi, P., & Naidu, V. (2017). Sense and avoid technology in unmanned aerial vehicles: A review. In *2017 International conference on electrical, electronics, communication, computer, and optimization techniques* (pp. 512–517). IEEE.

Chari, S., Gruen, D. M., Seneviratne, O., & McGuinness, D. L. (2020). Directions for explainable knowledge-enabled systems. (pp. 1–17). arXiv preprint arXiv:2003.07523.

Chari, S., Seneviratne, O., Gruen, D. M., Foreman, M. A., Das, A. K., & McGuinness, D. L. (2020). Explanation Ontology: A Model of Explanations for User-Centered AI. In J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, & L. Kagal (Eds.), *Lecture notes in computer science*: *vol. 12507*, *19th International semantic web conference, proceedings, Part II* (pp. 228–243). Athens, Greece: Springer.

Chuang, J.-H., & Ahuja, N. (1998). An analytically tractable potential field model of free space and its application in obstacle avoidance. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *28*(5), 729–736.

Cunningham, A., Wu, V., Biaz, S., & Jones, D. (2013). *Decentralized collision avoidance framework for unmanned aerial vehicles*: *Technical report CSSE 13-02*, (pp. 1–8). Auburn University.

Davies, J. T., & Wu, M. G. (2018). *Comparative analysis of ACAS-Xu and DAIDALUS detect-and-avoid systems*: *Technical report NASA-TM-2018-219773*, National Aeronautics and Space Administration.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors*, *37*(1), 32–64.

Falato, B. K. (1988). *National airspace system notice to airmen (NOTAM) system operational concept (NAS-SR-NOTAM)*: *Technical Report*, Atlantic City, NJ: Federal Aviation Administration Technical Center.

Fasano, G., Accardo, D., Tirri, A. E., & Moccia, A. (2016). Experimental analysis of onboard non-cooperative sense and avoid solutions based on radar, optical sensors, and data fusion. *IEEE Aerospace and Electronic Systems Magazine*, *31*(7), 6–14.

Fasano, G., Accardo, D., Tirri, A. E., Moccia, A., & De Lellis, E. (2015a). Challenges and solutions for vision-based sense and avoid. In *AIAA infotech @aerospace* (p. 0482). AIAA.

Fasano, G., Accardo, D., Tirri, A. E., Moccia, A., & De Lellis, E. (2015b). Radar/electro-optical data fusion for non-cooperative UAS sense and avoid. *Aerospace Science and Technology*, *46*, 436–450.

Feigenbaum, J., Jaggard, A. D., & Wright, R. N. (2011). Towards a formal model of accountability. In *Proceedings of the 2011 new security paradigms workshop* (pp. 45–56). ACM.

Feng, Y.-H., Teng, T.-H., & Tan, A.-H. (2009). Modelling situation awareness for context-aware decision support. *Expert Systems with Applications*, *36*(1), 455–463.

FLARM (2022). Traffic awareness and collision avoidance technology. https://flarm.com. (Accessed 12 May 2022).

Fu, Y., Ding, M., & Zhou, C. (2011). Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *42*(2), 511–526.

Fu, Y., Yu, X., & Zhang, Y. (2015). Sense and collision avoidance of unmanned aerial vehicles using Markov decision process and flatness approach. In *2015 IEEE international conference on information and automation* (pp. 714–719). IEEE.

Gao, W., Wang, K., Ding, W., Gao, F., Qin, T., & Shen, S. (2020). Autonomous aerial robot using dual-fisheye cameras. *Journal of Field Robotics*, *37*(4), 497–514.

Gellerman, N., Mullins, M., Foerster, K., & Kaabouch, N. (2018). Integration of a radar sensor into a sense-and-avoid payload for small UAS. In *2018 IEEE aerospace conference* (pp. 1–9). IEEE.

Graham, S., De Luca, J., Chen, W.-z., Kay, J., Deschenes, M., Weingarten, N., et al. (2011). Multiple intruder autonomous avoidance flight test. In *Infotech@ aerospace 2011* (p. 1420). AIAA.

Guarino, N. (1998). Formal ontology and information systems. In *Proceedings of the first formal ontology and information systems conference* (pp. 3–15). IOS Press.

Guarino, N., & Giaretta, P. (1995). Ontologies and knowledge bases: Towards a terminological clarification. In N. J. I. Mars (Ed.), *Towards very large knowledge bases: Knowledge building and knowledge sharing* (pp. 25–32). IOS Press.

Hall, D. L., & Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE*, *85*(1), 6–23.

Hein, G. W. (2000). From GPS and GLONASS via EGNOS to Galileo–positioning and navigation in the third millennium. *GPS Solutions*, *3*(4), 39–47.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S. (2009). *OWL 2 web ontology language primer*: *W3C recommendation*, World Wide Web Consortium.

Holdings, R. (2022). Drone box. https://www.redcatholdings.com/drone-box. (Accessed 12 May 2022).

Iris Automation (2022). Casia detect and avoid. https://www.irisonboard.com/casia/. (Accessed 12 May 2022).

Israelsen, J., Beall, M., Bareiss, D., Stuart, D., Keeney, E., & van den Berg, J. (2014). Automatic collision avoidance for manually tele-operated unmanned aerial vehicles. In *2014 IEEE international conference on robotics and automation* (pp. 6638–6643). IEEE.

Isufaj, R., Omeri, M., & Piera, M. A. (2022). Multi-UAV conflict resolution with graph convolutional reinforcement learning. *Applied Sciences*, *12*(2).

Jackson, J. A., Bošković, J. D., & Diel, D. (2015). Sensor fusion for sense and avoid for small UAS without ADS-B. In *2015 International conference on unmanned aircraft systems* (pp. 784–793). IEEE.

JARUS (2019). JARUS guidelines on specific operations risk assessment (SORA) V2.0. http://jarus-rpas.org/sites/jarus-rpas.org/files/jar_doc_06_jarus_sora_v2.0.pdf. (Accessed 12 May 2022).

Jenie, Y. I., Van Kampen, E. J., Ellerbroek, J., & Hoekstra, J. M. (2016). Taxonomy of conflict detection and resolution approaches for unmanned aerial vehicle in an integrated airspace. *IEEE Transactions on Intelligent Transportation Systems*, *18*(3), 558–567.

Jorris, T. R., & Cobb, R. G. (2008). Multiple method 2-D trajectory optimization satisfying waypoints and no-fly zone constraints. *Journal of Guidance, Control, and Dynamics*, *31*(3), 543–553.

Julisch, K., Suter, C., Woitalla, T., & Zimmermann, O. (2011). Compliance by design–bridging the chasm between auditors and IT architects. *Computers and Security*, *30*(6–7), 410–426.

Karelahti, J., Virtanen, K., & Öström, J. (2008). Automated generation of realistic near-optimal aircraft trajectories. *Journal of Guidance, Control, and Dynamics*, *31*(3), 674–688.

Karimi, J., & Pourtakdoust, S. H. (2013). Optimal maneuver-based motion planning over terrain and threats using a dynamic hybrid PSO algorithm. *Aerospace Science and Technology*, *26*(1), 60–71.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., et al. (2017). End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE international conference on computer vision* (pp. 66–75). IEEE.

Kim, Y., Gu, D.-W., & Postlethwaite, I. (2008). Real-time path planning with limited information for autonomous unmanned air vehicles. *Automatica*, *44*(3), 696–712.

Knublauch, H., Hendler, J. A., & Idehen, K. (2011). SPIN - Overview and Motivation. W3C Recommendation.

Koubaa, A., Allouch, A., Alajlan, M., Javed, Y., Belghith, A., & Khalgui, M. (2019). Micro air vehicle link (MAVLink) in a Nutshell: A survey. *IEEE Access*, *7*, 87658–87680.

Ladetto, Q., & Merminod, B. (2002). Digital magnetic compass and gyroscope integration for pedestrian navigation. In *9th International conference on integrated navigation systems, St-Petersburg* (pp. 27–29). International Conference on Integrated Navigation Systems.

Langejan, T., Sunil, E., Ellerbroek, J., & Hoekstra, J. (2016). Effect of ADS-B characteristics on airborne conflict detection and resolution. In *6th SESAR innovation days* (pp. 1–8).

Lassila, O., & Swick, R. R. (1999). *Resource description framework (RDF) model and syntax specification*: *W3C recommendation*, W3C.

Lieb, J., & Volkert, A. (2020). Unmanned aircraft systems traffic management: A comparsion on the FAA UTM and the European CORUS ConOps based on U-Space. In *2020 AIAA/IEEE 39th digital avionics systems conference* (pp. 1–6). IEEE.

Lin, Y., & Saripalli, S. (2015). Sense and avoid for unmanned aerial vehicles using ADS-B. In *2015 IEEE international conference on robotics and automation* (pp. 6402–6407). IEEE.

Martín-Lammerding, D., Astrain, J. J., & Córdoba, A. (2021). A reference ontology for collision avoidance systems and accountability. In *SEMAPRO 2021, the fifteenth international conference on advances in semantic processing* (pp. 22–28). IARIA.

Martín-Lammerding, D., Astrain, J. J., & Córdoba, A. (2022a). A multi-UAS simulator for high density air traffic scenarios. In *the eleventh international conference on advances in vehicular systems, technologies and applications VEHICULAR 2022* (pp. 32–37). IARIA.

Martín-Lammerding, D., Astrain, J. J., & Córdoba, A. (2022b). Dronetology-cas, anti-collision ontology for UAS. https://dronetology.net/dronetology-cas (Accessed 12 May 2022).

Martín-Lammerding, D., Astrain, J. J., & Córdoba, A. (2022c). Dronetology, the UAS ontology. https://dronetology.net/dronetology. (Accessed 12 May 2022).

McCallie, D., Butts, J., & Mills, R. (2011). Security analysis of the ADS-B implementation in the next generation air transportation system. *International Journal of Critical Infrastructure Protection*, *4*(2), 78–87.

Muraru, A. (2011). A critical analysis of sense and avoid technologies for modern UAVs. In *2011 International conference on mechanical, industrial, and manufacturing engineering* (pp. 162–165).

Nikolos, I. K., Valavanis, K. P., Tsourveloudis, N. C., & Kostaras, A. N. (2003). Evolutionary algorithm based offline/online path planner for UAV navigation. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *33*(6), 898–912.

Noy, N. F., & McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology*: *Technical report*, Stanford Knowledge Systems Laboratory.

Ong, H. Y., & Kochenderfer, M. J. (2017). Markov decision process-based distributed conflict resolution for drone air traffic management. *Journal of Guidance, Control, and Dynamics*, *40*(1), 69–80.

Owen, M. P., Panken, A., Moss, R., Alvarez, L., & Leeper, C. (2019). ACAS Xu: Integrated collision avoidance and detect and avoid capability for UAS. In *2019 IEEE/AIAA 38th digital avionics systems conference* (pp. 1–10). IEEE.

Papa, U. (2018). Sonar sensor model for safe landing and obstacle detection. In *Embedded platforms for UAS landing path and obstacle detection* (pp. 13–28). Springer.

Papa, U., Ariante, G., & Del Core, G. (2018). UAS aided landing and obstacle detection through LIDAR-Sonar data. In *2018 5th IEEE International workshop on metrology for aerospace* (pp. 478–483). IEEE.

Peach, N. (1995). Bearings-only tracking using a set of range-parameterised extended Kalman filters. *IEE Proceedings D (Control Theory and Applications)*, *142*(1), 73–80.

Prud'hommeaux, E., & Seaborne, A. (2008). SPARQL Query language for RDF. W3C Recommendation.

PX4 (2022). Open source autopilot. https://docs.px4.io/. (Accessed 12 May 2022).

Ramasamy, S., Sabatini, R., & Gardi, A. (2014). Avionics sensor fusion for small size unmanned aircraft sense-and-avoid. In *2014 IEEE metrology for aerospace* (pp. 271–276). IEEE.

Raspberry Pi Foundation (2022). Raspberry pi3. https://www.raspberrypi.org/. (Accessed 12 May 2022).

Richards, A., & How, J. P. (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American control conference (IEEE Cat. No. CH37301), Vol. 3* (pp. 1936–1941). IEEE.

Sánchez, P., Casado, R., & Bermúdez, A. (2020). Real-time collision-free navigation of multiple UAVs based on bounding boxes. *Electronics*, *9*(10), 1632.

Shakernia, O., Chen, W.-Z., & Raska, V. (2005). Passive ranging for UAV sense and avoid applications. In *Infotech@ aerospace* (p. 7179). AIAA.

Sislak, D., Rehak, M., Pechoucek, M., Pavlicek, D., & Uller, M. (2006). Negotiation-based approach to unmanned aerial vehicles. In *IEEE workshop on distributed intelligent systems: Collective intelligence and its applications* (pp. 279–284). IEEE.

Stanford University (2022). Protege. https://protege.stanford.edu/. (Accessed 12 May 2022).

Strobel, A., & Schwarzbach, M. (2014). Cooperative sense and avoid: Implementation in simulation and real world for small unmanned aerial vehicles. In *2014 International conference on unmanned aircraft systems* (pp. 1253–1258). IEEE.

Sunberg, Z. N., Kochenderfer, M. J., & Pavone, M. (2016). Optimized and trusted collision avoidance for unmanned aerial vehicles using approximate dynamic programming. In *2016 IEEE international conference on robotics and automation* (pp. 1455–1461). IEEE.

Tl-Elektronic (2022). Tl black box. https://www.tl-elektronic.com/. (Accessed 12 ay 2022).

UAV Navigation (2022). Flight data recorder. https://www.uavnavigation.com/. (Accessed 12 May 2022).

Van Den Berg, J., Guy, S. J., Lin, M., & Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research* (pp. 3–19). Springer.

Vera, S., Cobano, J. A., Alejo, D., Heredia, G., & Ollero, A. (2015). Optimal conflict resolution for multiple UAVs using pseudospectral collocation. In *2015 23rd Mediterranean conference on control and automation* (pp. 28–35). IEEE.

Watts, A. C., Ambrosia, V. G., & Hinkley, E. A. (2012). Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use. *Remote Sensing*, *4*(6), 1671–1692.

Williamson, T., & Spencer, N. A. (1989). Development and operation of the traffic alert and collision avoidance system (TCAS). *Proceedings of the IEEE*, *77*(11), 1735–1744.

Wing, M. G., Eklund, A., & Kellogg, L. D. (2005). Consumer-grade global positioning system (GPS) accuracy and reliability. *Journal of Forestry*, *103*(4), 169–173.

Zarandy, A., Zsedrovits, T., Pencz, B., Nameth, M., & Vanek, B. (2015). A novel algorithm for distant aircraft detection. In *2015 International conference on unmanned aircraft systems* (pp. 774–783). IEEE.

Zhao, Y., Jiao, L., Zhou, R., & Zhang, J. (2017). UAV formation control with obstacle avoidance using improved artificial potential fields. In *2017 36th Chinese control conference* (pp. 6219–6224). IEEE.

Zuehlke, D., Prabhakar, N., Clark, M., Henderson, T., & Prazenica, R. J. (2019). Vision-based object detection and proportional navigation for UAS collision avoidance. In *AIAA scitech 2019 forum* (p. 0960). AIAA.