Article

Automated Conflict Management Framework Development for Autonomous Aerial and Ground Vehicles

David Sziroczák * and Daniel Rohács

Department of Aeronautics and Naval Architecture, Budapest University of Technology and Economics, 1111 Budapest, Hungary; rohacs.daniel@kjk.bme.hu
* Correspondence: sziroczak.david@kjk.bme.hu

Abstract: The number of aerial- and ground-based unmanned vehicles and operations is expected to significantly expand in the near future. While aviation traditionally has an excellent safety record in managing conflicts, the current approaches will not be able to provide safe and efficient operations in the future. This paper presents the development of a novel framework integrating autonomous aerial and ground vehicles to facilitate short- and mid-term tactical conflict management. The methodology presents the development of a modular web service framework to develop new conflict management algorithms. This new framework is aimed at managing urban and peri-urban traffic of unmanned ground vehicles and assisting the introduction of urban air mobility into the same framework. A set of high-level system requirements is defined. The incremental development of two versions of the system prototype is presented. The discussions highlight the lessons learnt while implementing and testing the conflict management system and the introduced version of the stop-and-go resolution algorithm and defines the identified future development directions. Operation of the system was successfully demonstrated using real hardware. The developed framework implements short- and mid-term conflict management methodologies in a safe, resource efficient and scalable manner and can be used for the further development and the evaluation of various methods integrating aerial- and ground-based autonomous vehicles.

Keywords: autonomous conflict management; UTM; UAV; UGV; U-Space; framework development

check for updates

Citation: Sziroczák, D.; Rohács, D. Automated Conflict Management Framework Development for Autonomous Aerial and Ground Vehicles. *Energies* **2021**, *14*, 8344. https://doi.org/10.3390/en14248344

Academic Editor: Arno Eichberger

Received: 9 November 2021
Accepted: 8 December 2021
Published: 10 December 2021

1. Introduction

The use of autonomous vehicles, both road going and aerial, is expanding rapidly these days. Just in the case of unmanned aerial vehicles, between 2016 and 2021 Goldman Sachs [1] predicted a USD 100 billion market opportunity, with a further 16–24% annual growth predicted between 2026–2028 [2,3]. The market opportunities for autonomous road vehicles are no less significant. To serve a market of this size, a significant amount of traffic needs to be managed. In controlled airspaces, by 2050 EUROCONTROL predicts that the total flight hours of UAS (unmanned aerial systems) in Europe will account for 20% of total traffic, an estimated 7 million flight hours [4]. Higher demand will be seen at the very low level (VLL) airspace, below 120 m (400 ft) AGL (above ground level), often referred to as the U-Space in the EU. In U-Space, about 250 million commercial flight hours are predicted in urban environments, 20 million in rural settings and 80 million for hobby use. This demand is a magnitude higher than what the current manned air traffic control systems handle.

In the case of road vehicles, as there is no central control, the system relies on individual drivers to manage conflict situations. In the EU, the ACEA (European Automobile Manufacturers Association, Brussels, Belgium) estimates 268 million road vehicles in 2018, with an average growth rate of 2% compared to the previous year [5]. It is further predicted by the Victoria Transport Policy Institute (Victoria, BC, Canada) that by 2045 half of new vehicles will be autonomous, and by 2060 half of the active vehicle fleet will be autonomous [6]. Based on the defined autonomous driving levels [7] autonomous road vehicles are designed to achieve safe and efficient driving individually, and thus control and conflict management of these vehicles could also benefit road safety. Based on the high volume of traffic to be managed, it is predicted that a system similar to today's air traffic control will not be sufficient for these vehicles either. Current ATC (air traffic control) procedures are human labour intensive and offer poor automation possibilities.

Aircraft traffic management for unmanned systems, or UTM in short, is an active research field today; both national and international organizations actively develop and test various concepts for the management of this novel class of airspace users. ICAO has already published the third edition of their UTM harmonization guidelines [8]. The document is aimed at providing a common framework along which nations can develop harmonised solutions for UTM activities. In addition to working towards international compatibility, the initiative also helps to reduce the costs associated with the development of UTM solutions. Essentially, all nations have to find solutions for this novel challenge, and it would be beneficial if the development activities were shared in some form as well. In the USA both NASA (National Aeronautics and Space Administration, Washington, DC, USA) and the FAA (Federal Aviation Administration, Washington, DC, USA) are actively developing the nation's Next Generation Air Transportation System, planned to be in operation by 2025. While the system is aimed at improving various aspects of airspace use, integrating UAS is a significant part of the challenge. Both organizations have been actively engaged in the UAS Integration Pilot Program (IPP) (ended in 2020, now continued in the BEYOND programme) and the UTM Pilot Program (UPP) [9] since 2019, which are aimed at integrating the current estimated 350,000 UAS and future systems into the national airspace. ASTM (American Society for Testing and Materials, West Conshohocken, PA, USA) has developed standards for the remote identification of UAS, published as ASTM F3411 [10]. In the EU, the development of the U-Space system (part of the SESAR programme) is aimed at implementing UTM solutions [11]. At the national level, but also as part of the U-Space initiative, UTM systems have already been introduced, for example, in Finland (GOF, Gulf of Finland [12]), Switzerland (SUSI, Swiss U-Space Implementation, Zug, Switzerland, [13]) and the UK (CAA Innovation Hub, London, UK, [14]). There is also activity at the private level; companies such as Altitude Angel, Skyguide, AirMap and sees.ai have already developed UTM solutions, and some of them are already providing these solutions today. There are also various consortia, such as DOMUS, USIS, DIODE, EuroDRONE, SAFIR, VUTURA, GeoSAFE, PODIUM, SAFEDRONE and CORUS developing various aspects of UTM and related activities. In Hungary, where the research presented was performed, HungaroControl has founded the UTM Innovation HUB and CybAIR Cluster, which are both contributing towards UTM systems development.

In the autonomous ground vehicle (often referred to as self-driving car) industry primary emphasis is placed on collision avoidance systems in individual vehicles. As was previously mentioned, autonomous operation is being developed as a stepped approach, and many of these concepts also appear as driver assistance systems in conventional ground vehicles. These systems include collision warning systems (front and rear) adaptive cruise control, lane assist, road sign detection and similar systems. Approaches such as smart city or more generally smart mobility are aimed at increasing the safety, sustainability and efficiency of current traffic systems, but they do not necessarily rely on the control of vehicles. Solutions include the analysis and evaluation of traffic, variable speed limits, dynamic traffic signal timing or dynamic lanes. In the case of cooperating and connected vehicles, such systems could provide smart routing and speed control for optimum traffic efficiency, essentially reducing time spent in congestions and improving travel experience for drivers and passengers.

This paper is aimed at developing a system prototype that can be used to implement the short- and mid-term conflict management of a local U-Space system containing both autonomous aerial and ground vehicles, along with other potential users. The inclusion of both aerial and ground autonomous systems in a common management framework can be considered as a novel idea. The pairing of self-driving cars and UAS has been experimented with, but only for specific applications such as food delivery [15], search and rescue [16] and infrastructure inspection [17]. There are also research works available that focus on the technological specifics of such integration [18,19]. Published studies of conflict management approaches that combine autonomous ground and aerial vehicles are unknown to the author.

As the whole concept of unmanned traffic management, both for UGV and UAV, is new, there are no set standards or solutions available today, only development initiatives. The only regulation available in this field is that UGV must abide by the applicable highway code of a given country in order to be compatible with current (manned) traffic. In the case of UAV, there are regulations concerning the individual use of vehicles (such as the EASA or FAA regulations), but there are no standards regarding the urban environment, or any sort of other traffic regulation for that matter, as this type of dense traffic simply does not exist today. As a result, systems being developed today do not need to be prepared to comply with any particular standard; rather, they can form the base of developing standards and regulations for the future.

While the concept of interaction between aerial and ground can be questionable, situations do arise where they can come into conflict. The first category is related to take-off and landing type manoeuvres, in which case UAS voluntarily moves close to the ground to where it can be affected by ground traffic. A typical use of this would be collection and delivery type scenarios. Additionally, this category includes vehicle–aircraft interactions, such as when a drone takes off or lands on the top of a delivery van, which can be either static or moving. The second category includes involuntary near-ground activities, such as emergency landing or descent due to failure, weather effects or while evading other airspace users or obstacles. The third category relates to environmental effects in man-made environments. Concepts such as urban canyons [20], essentially artificial channels that have a significant local impact on wind and gust characteristics (also on GPS and communications), are already known to researchers studying built environments. Additionally, large vehicles, such as buses, HGVs or trains can generate significant air flow in a constrained environment, which is often referred to as the piston effect. These effects can be significant enough that UAS would have to consider the presence, movement and potential path of ground vehicles even when operating at altitudes where direct physical contact is of no concern.

Developing a conflict management system involving both unmanned aerial and ground vehicles is essential for the advancement of autonomous vehicle use, especially in urban or peri-urban environments. In the field of aviation, the use of vehicles in an urban environment is referred to as urban air mobility, or UAM for short. Systems being developed for autonomous ground vehicles utilise sensors that primarily detect in the ground plane, as that is where the road traffic is expected. As a result, they are not equipped to detect and deal with aerospace traffic. Based on the roadmaps and published research available today, this is not likely to change in the near future. In the case of aerospace vehicles, as it was mentioned, current ATC procedures are not applicable, mainly due to the performance constraints and poor automation opportunities of current procedures. They key capability such a combined traffic system must provide are the following:

- Adequate level of safety: system must eliminate (significantly reduce) number and severity of accidents.
- Treatment of a wide range of users: ground and air vehicles must be included, along with potential other users (pedestrians, obstacles, etc.).
- High performance: management of conflicts must not significantly affect the mission times of the users or result in unacceptably long detours.
- Acceptable cost of operation: where cost includes system development, installation, maintenance, labour and cost to system users.

Automation in conflict management is the only possibility to provide the required capabilities, mainly due to the high number of users and conflicts envisioned. As such, a common and automated treatment for both classes of vehicles is essential. This paper presents a prototype solution developed to provide adequate capability for the management of conflicts between these users.

The first part of this paper deals with the methodology used to define the requirements and key components of the proposed conflict management system. It begins with an introduction to conflict management concepts and their applicability to the defined purpose. Then, high-level requirements are defined for the proposed conflict management system. In the results section, the system architecture of the implemented conflict management system is presented, based on two iterations and live testing of the system prototype. The discussion chapter presents the results and insights gained from the testing of the prototype system and proposes further development directions.

2. Materials and Methods

2.1. Conflict Management Methodologies

The proposed conflict management system implements a short- and mid-term tactical conflict management methodology. Conflict management approaches can be generally divided into 2 categories: strategic and tactical management. The basis of the categorization relates to the time factor of conflict detection and the available means to manage the event. The discussion of conflict management in this chapter is primarily based on concepts found in the field of aviation.

Strategic management refers to measures taken before the actual operation has begun. The activities performed during strategic management are aimed at minimizing the potential effects arising from the lack of information or faulty planning of routes. In aviation, typical planning issues include the availability of airspace (including routing through restricted and forbidden airspace), lack of air traffic control capacity (overburdened control officers) and flight plans received from other users. In all these cases, the conflict situation can be predicted in advance of the flight operation. The usual tools for strategic conflict management are the rerouting of flight paths or rescheduling of the operations, or when required, even cancelling the flight altogether. As an addendum, considering from a more abstract perspective, the long-term infrastructure planning and traffic policy making can also be considered as strategic level conflict management, just at the timescale of many years. As an example, NTU (Nanyang Technological University) Singapore proposed airspace design methods for UAV traffic in an urban environment, including vertical separation, "traffic light effect", digital unidirectional lanes and other features [21]. In the future, implementing these designs could contribute towards minimizing the situations where tactical level conflict arises. The longer-term strategy also has relevance regarding the infrastructure available to detect users and conflicts during operations (the tactical level). The system proposed in this research does not implement strategic conflict management, and as such it will not be discussed further.

Tactical conflict management is responsible for preventing collisions in the following situations:

- Between aircraft and other aircraft;
- Between aircraft and ground;
- Between aircraft and airspace (or geofence);
- Between aircraft and obstacles.

As these conflict situations arise during operation, most of the time they are not predictable at the planning and path approval stages. As a result, the time available to detect and resolve these conflict situations is significantly shorter than in the case of strategic management. The management of conflict situations usually requires active commands, which will result in deviation from the planned path for at least one user involved in the conflict. In the world of manned aviation these commands indicate a change in direction, altitude or speed. The command can come from 3 sources: ground-based systems (voice command from air traffic control officer through radio), on-board systems (TCAS—Traffic Collision Avoidance System—electronic indicators on flight display) or conflict analysis and decision from the pilot itself.

In the case of tactical management, the most critical factor is time. For human pilots (or drivers), processing information and coming to decisions is directly related to the time taken to do so. In the case of urban environments, the typical distances involved in conflict situations are generally small, which leaves little time for human decision making.

The tactical conflict management can be divided into 2 stages: conflict detection and conflict resolution. Detection methods can be grouped into 3 categories: deterministic, probabilistic and worst-case methods. While each of the detection methods have advantages and disadvantages compared to each other, their applicability is only valid in the short term, regardless of the method. An overview and classification of published classic conflict detection methods is presented in Table 1.

Table 1. Conflict management methods overview.

Type	Conflict Management Methods/Areas Addressed	Author and Reference
Deterministic	Optimal avoidance manoeuvres, conflict zones	Krozel et al. [22]
	Expert system for avoidance manoeuvres	Iijima et al. [23]
	Rule-based conflict management	Coenen et al. [24]
	Aircraft ground collision	GPWS [25]
	Manoeuvres	Bilimoria et al. [26]
	Optimal tactical and strategic manoeuvres	Krozel and Peters [27]
	Generalised conflict zones	Havel and Husarcik [28]
	Aerial collision avoidance	TCAS [29]
	Conflict alert evaluation	Ford [30]
Worst case	Optimal manoeuvres	Tomlin et al. [31]
	Uncertainty of planned flightpaths	Shepard et al. [32]
	Worst case turns or velocity changes	Shewchun and Feron [33]
	TCAS system limitations	Ratcliffe [34]
Probabilistic	Rapid conflict prediction	Paielli and Erzberger [35]
	Probability-based detection	Carpenter and Kuchar [36]
	Markov chain-based probability estimation	Bakker and Blom [37]
	Trajectory confidence model	Williams [38]
Hybrid	Genetic algorithm-based avoidance	Durand et al. [39]
	Stochastic hybrid model including wind effects	Glover and Lygeros [40]
	Monte Carlo simulation based	Visintini et al. [41]
	Non-cooperating target classification	Palme et al. [42]

In terms of conflict resolution, the earlier a conflict is detected and acted on, the more efficient and safer the conflict resolution will be. The conflict management approach studied in this paper focuses on short- and mid-term tactical management and as such, other methods will not be discussed further. For the sake of completeness, Table 2 presents the compiled overview of generalised approaches to conflict management, as defined by the researchers in this study. The time considered spans the complete time horizon of an impact event, including the possible very long term preceding activities and the post-impact treatment, which are not classically included in conflict management. The definitions and boundaries of the various levels can vary between researchers; the ones presented here represent the definitions used in this study and by other research activities by the researchers. Here, impact refers to the potential consequence of not managing the conflict, e.g., it can be actual physical impact between vehicles or environment or damage due to abrupt manoeuvres.

Table 2. Generalised conflict management timeline.

Time Scale Relative to Impact	Conflict Management Level	Aim of Activities	Available Tools
Up to about 10 years before	Strategic level system and technology planning	Develop safe and efficient procedures, integrate new technologies	Policy making, Research and development
Up to about 5–10 years before	Strategic level infrastructure planning	Deploy and develop infrastructure both ground based and onboard systems	City planning, infrastructure planning, vehicle design codes
Up to about 1–5 years before	Strategic level traffic planning	Determine and influence modes and volume of traffic	Policies, incentives, regulations, market development
Hours before	Strategic conflict management	Route planning	Rerouting, rescheduling, cancelling operations
5–10 s before	Tactical conflict management (mid-term)	Detection and alternative path planning	Active trajectory change: direction or speed adjustment
1–5 s before	Tactical conflict management (short-term)	Attempt to avoid impact	Evasive manoeuvres
Fraction of second before	Active safety systems	Preventive steps to minimise impact	Passenger safety systems, seat belt tensioners, airbags, etc.
During impact	Passive safety systems	Minimise impact	Crashworthy structures, optimal occupant positioning, etc.
Post impact	Post-impact treatment	Recover from effects of impact	Emergency services, warning for other traffic

When dealing with short- and mid-term conflict management in the case of autonomous vehicles, challenges arise in the resolution step. None of the current methods relying on human awareness or accepting and effecting commands are appropriate in this case, reducing the options available to digital commands either from a ground-based or on-board computational system.

2.2. *High-Level Conflict Management System Requirements*

The following chapter discusses the core principles and the high-level requirements that were defined to develop the conflict management system.

2.2.1. Performance Requirements

The proposed conflict management system is aimed at serving a wide variety of users, both aerial and ground based, implementing short- and mid-term tactical conflict management solutions. As such, the system needs to be able to detect conflicts and issue appropriate commands within the available time frame, which is in the order of 10 s or less. Due to the high magnitude of unmanned traffic predicted in the future, all steps of the conflict management process need to work autonomously, without human interaction.

2.2.2. Technology and Solution Independence

In addition to the high number of operations and users, it is also predicted that a diverse variety of users will need to be served. While there are initiatives that are (at least in theory) standardised in the automotive world (OBD—on board diagnostics—connectors for example) it is very unlikely that autonomous capabilities and communication solutions developed by individual manufacturers will be standardised in the near term. The case is similar in the UAS industry. There are initiatives, mostly in the open-source world, to standardise communication and control protocols, for example the MAVLink (micro air vehicle link) protocol. Individual manufacturers, however, still mostly rely on proprietary closed solutions, where even different models from the same manufacturer might be using different communication and control systems. This is the case for DJI for example, the most widespread civilian-use COTS (commercial off-the-shelf) drone solution. It is unknown whether any one particular technology will be adopted in the future as a standard solution, and at the moment there seem to be no initiatives either. Regarding communication solutions, there are also no dominant approaches; however, V2X (vehicle-to-anything) communication methods are rapidly changing active research field today. Potential solutions include mobile network-based (4G/LTE, 5G), LORA, WLAN, Bluetooth and other technologies in use today. While 5G solutions have been demonstrated, linked to this project [43], and it is likely that mobile network-based solutions will become more and more widespread in the future, it cannot be said with certainty that it will become the definite solution in the future. As such, the conflict management system needs to be technology and solution independent.

2.2.3. Wide Variety of System Users and Components

The system needs to accommodate a wide range of users. In addition to the technological aspects already discussed, the specific type of users also represents a wide range. In case of UAVs, the smaller multi-copter designs are of primary concern, but VTOL (Vertical Take-off and Landing—either small or aerial taxi sized) and potentially fixed-wing aircraft and others (balloons, airships, etc.) could be present in the airspace. In terms of ground-based users, in addition to self-driving cars, manned vehicles, cyclists and pedestrians can also be present in a traffic situation. The environment and obstacles can be regarded as a third source of "users" in a conflict management system, as impact with these elements must also be avoided. As such, from the planning stage the system is aimed at integrating data from GIS (Geographic Information System) systems, including Geofence definitions.

2.2.4. No Controller Development

A very important criterium for the system is that the individual users must be able to operate using their own control algorithms. That practically means all hardware must work in a plug and play manner. It is infeasible to rely on the conflict management system for low-level control for 2 reasons. For one, it would involve extensive development for each vehicle as a dynamic system, not even considering payload, centre of gravity and other configuration possibilities. This is an activity that the provider of a conflict management system cannot afford to do. Another aspect is responsibility. The conflict management system must provide a command to follow to avoid the conflict, but it should not decide how to follow the command. Individual control tuning, dynamics, limitations and assistance systems (collision prevention, obstacle detection, lane assist, etc.) need to function individually and in addition to the conflict management system. Some, such as the assistance systems or geofencing, essentially act as additional safety barriers for the management of conflicts. The same logic applies to mission planning software. For UAV these tools are referred to as ground control stations (GCS or GC). All mission (autonomous flight)-capable UAVs have some form of GCS for the operator to setup the desired mission. The concept is very similar for UGV, but the mission planning software will likely appear as a map application integrated into the dashboard or similar. For UAV the open-source world (PX4 and ArduPilot are the most common control software) has well-established solutions (QGroundControl, MissionPlanner or APM), and other manufacturers use their own solutions (DJI Pilot, for example). There are also dedicated commercial solutions aimed mainly at commercial and enterprise level users, such as UGCS or Auterion Mission Control. These GCS software must be used by the operator, as the responsibility of planning and executing their operations cannot be taken up by the conflict management system provider.

2.2.5. Modular Software Solution

The software solution needs to have modular architecture. This enables the integration of the various types of users by developing type-specific communication interfaces for a given solution (MAVLink, DJI SDK, etc.). This also futureproofs the system as upcoming new types can be included by adding new modules to connect them. The conflict management algorithms also benefit from the modular architecture, as different types of detection and resolution methods can be developed as separate modules and their performance tested. The development process follows LEAN software development principles, aiming to minimise the time of each plan–build–test–evaluate cycle when new modules and functionality are implemented. To achieve this, it is important to start the development with the most widespread and available technologies, solutions and software stacks.

2.2.6. XITL Tools Integration

XITL [44] (X = anything in the loop) integration is a generalisation of using testing tools such as HITL (hardware), SITL (software), VITL (vehicle) and so on. This enables the development, testing and even operation of systems using a wide variety of sources, real world or simulated. Advanced concepts such as digital twins [45] or scenarios in the loop [43] have already been demonstrated for UGV as part of the Hungarian Autonomous Systems National Laboratory, in cooperation with the ZalaZone Automotive Proving Ground facility [46,47]. Simulation tools are especially beneficial in the case of conflict management, primarily because the consequences of mismanaging the conflict are minimal as opposed to physical impact between real-world vehicles. There are also additional benefits in terms of development time, cost, flexibility, safety, security and also convenience. While UGV operations can also be affected, especially in the case of UAVs, simulation tools also remove constraints arising from weather. Another major benefit is that significantly more data can be collected or generated from simulations, with more accuracy; measurement does not disrupt the phenomenon to be measured, and there is no measurement of uncertainty and noise. As an added benefit it is very easy to generate "fake" data programmatically, which can be used to test situations that would be difficult to orchestrate using real hardware or even high-fidelity simulators. The proposed conflict management tool needs to be designed so that it can integrate with XITL tools seamlessly.

2.2.7. Summary of Requirements

In summary, the key requirements towards proposed system are:

- Detecting and resolving tactical conflicts in an approximately 10 s timeframe;
- Fully autonomous operation;
- Technology and solution independent;
- Wide range of system users integrated;
- Plug and play integration;
- Modular software solution;
- XITL tools integration.

Figure 1 shows the architecture of the proposed framework concept. The key development areas are coloured in blue; these are the components that need to be built from scratch to achieve the desired functionality. The communication and user layers are items that are intentionally left to use only COTS solutions. The detection and resolution methods are coloured differently. The framework can work by integrating these algorithms, even if they are already developed or available only as a black box tool. The tools used can be changed in a modular way in the system. The final piece of the system is the GUI (graphical user interface), which enables a human supervisor to monitor the system and interact with it if necessary. It is not coloured blue, because there is no strict need to develop an independent GUI; the proposed system could integrate with other tools with existing GUIs, as long as appropriate interfaces are developed for data sharing.

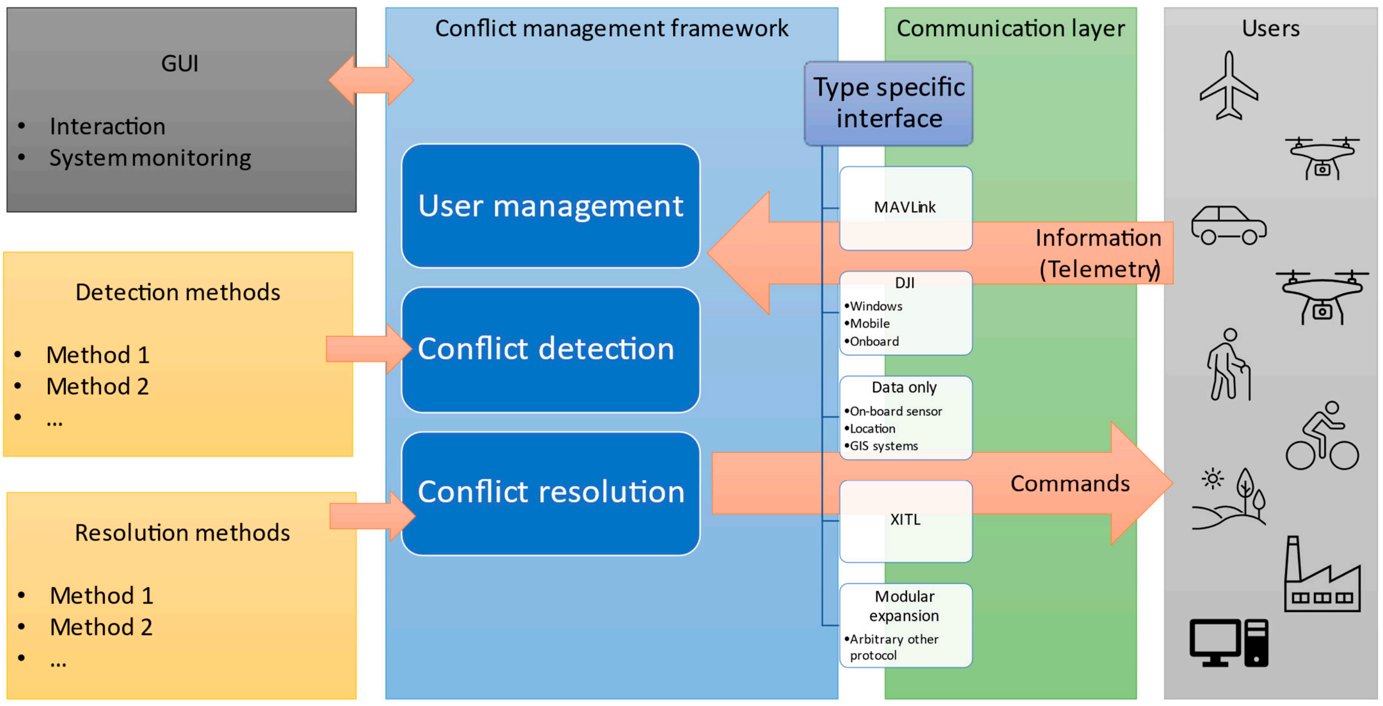

Figure 1. Conflict management system high level architecture.

3. Results

This chapter discusses the software implementation of the proposed conflict management framework, and the results achieved testing the framework with simulation and real hardware. Up to this point in the research project two versions of the framework were implemented, both of which will be presented along with the reasons for the development of the second iteration.

The programming language chosen for the system is Python (v3.8), as it is a language well suited for prototyping, and a wide variety of modules are available both for IT and engineering purposes. In the following, "system" always refers to the implemented conflict management system, unless otherwise noted.

3.1. First Version of Software Implementation

3.1.1. Overview

The first version of the conflict management software was developed as a self-contained computer application, where the conflict management framework is started at program execution, and the system users' missions were also started from the application. Essentially, this would enable the system to provide conflict management for pre-defined scenarios for testing purposes. Note that this is not equivalent to a scripted scenario, as the users are only given commands to begin their missions, after which the system only manages the conflicts, not the individual paths.

Figure 2 shows a top-level overview of the first framework version's workflow. First, in the system initialization step, the system goes through five tasks, after which it initiates an infinite loop where conflict management is facilitated. The system components are presented and discussed following the order presented in the flow diagram.

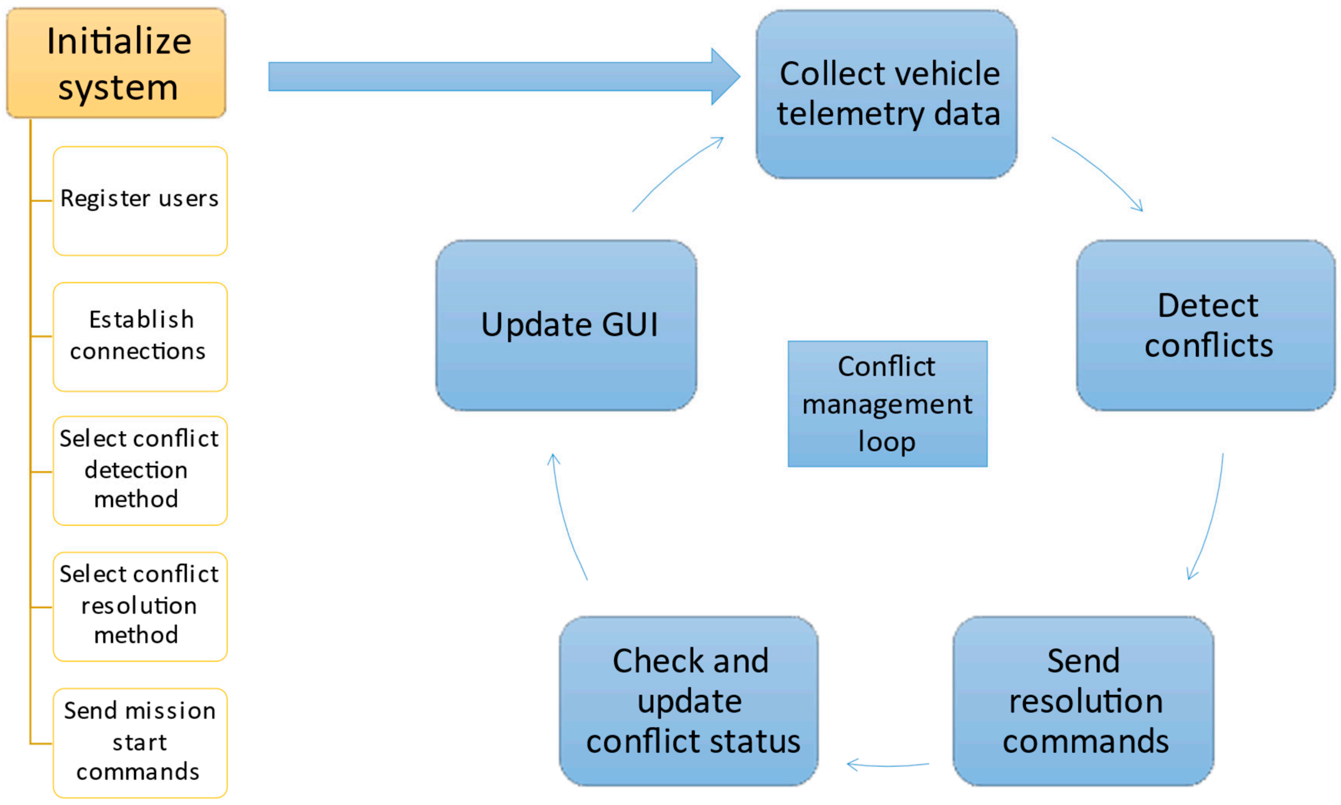
Initialize system
Register users
Establish connections
Select conflict detection method
Select conflict resolution method
Send mission start commands
Collect vehicle telemetry data
Detect conflicts
Send resolution commands
Check and update conflict status
Update GUI
Conflict management loop

Figure 2. Top level flow diagram of first framework version.

3.1.2. System Users and Connections

The users of the system are the individual vehicles, people, etc., for which the system provides the conflict management service. Following the defined requirements and key principles, the users are represented as an abstract class, from which the individual user types are inherited, thus ensuring modularity. The first version implements the following user types:

- Algorithmically generated (“fake”) location data;
- MAVLink connection:
  - SITL simulation
  - Real hardware.

The algorithmically generated data essentially calculates a time series of position data, and provides it for the conflict management system, without considering the actual vehicle type or its dynamics. Additionally, the vehicle receives no control commands from the system. This class of users represents a vehicle, which is not a cooperative member of the conflict management framework; however, its position is known. For example, positions could be received through protocols and channels not implemented in the system yet, or its position could be inferred with using computer vision or similar methods and the position provided by other users of the system. In this project, the class was initially implemented to enable the research group’s self-driving smart car (Figure 3) to integrate into the system.

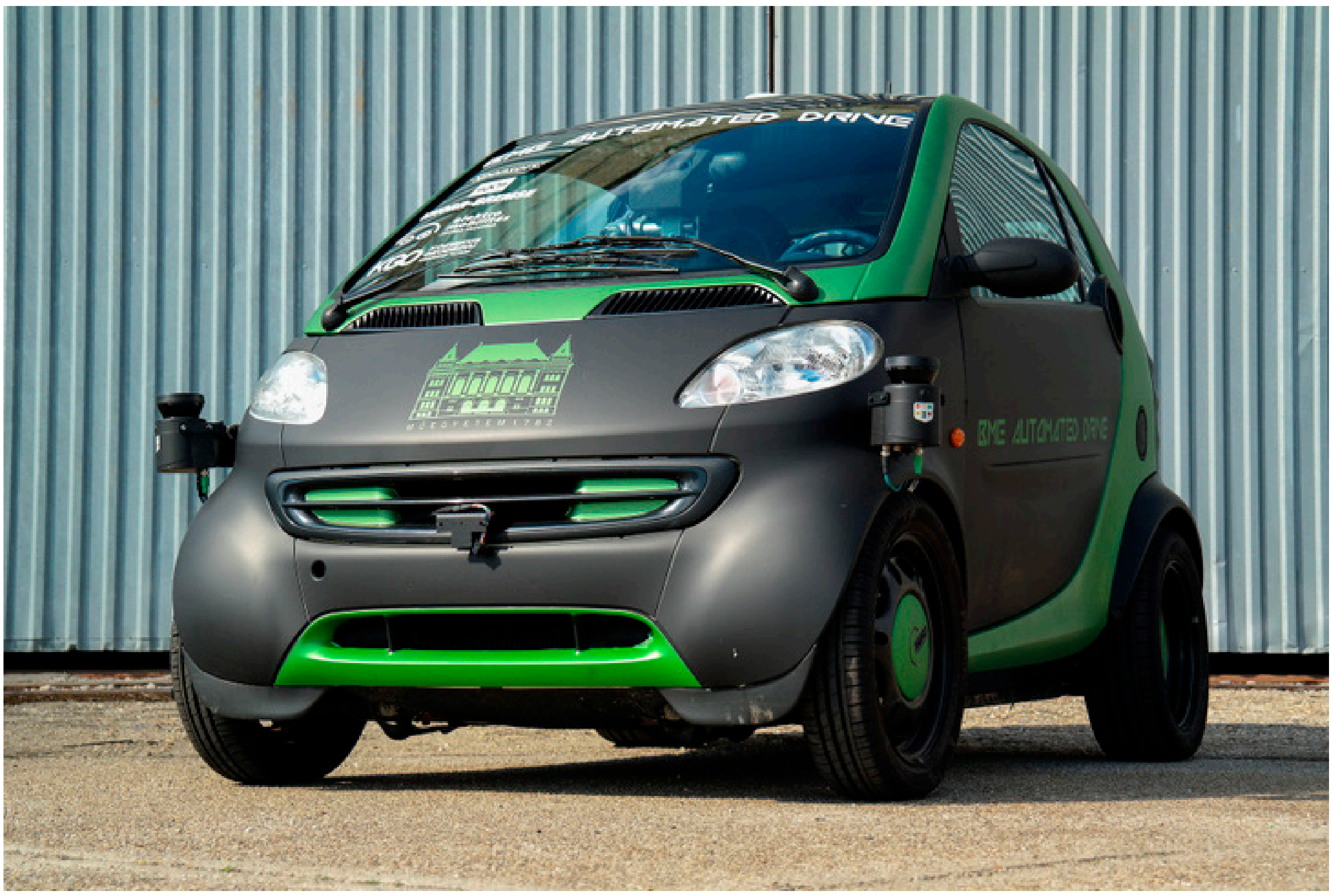

Figure 3. BME Automated Drive self-driving Smart [48].

The Smart currently operates as a self-contained unit; it is able to follow pre-specified paths in an autonomous way. However, with its current implementation, it cannot receive commands from an off-board source, nor does it provide telemetry data in any standardised format. As such, to integrate it into the system, the Smart provides its current position along the pre-defined path as simple coordinates data, which conceptually works the same way as the algorithmically generated data from the system's point of view. The algorithmically generated class is also very useful to test specific scenarios, as actual vehicle control and dynamics do not constrain the possible timings and paths.

The MAVLink connection class implements two-way communication between MAVLink-capable vehicles and the system. There are three options available in the system:

- PX4 SITL simulator;
- X500 UAV with PX4 flight control;
- "PX4 in a box" unit.

PX4 [49] is an open-source control software that is ported to a wide variety of hardware. It is capable of controlling fixed-wing and multi-rotor UAVs and UGVs among many other configurations. The PX4 SITL simulator can be run on any compatible computer (Windows, Linux, Mac), with relatively low resource requirements. The SITL itself runs the same flight controller algorithms that are used when flying actual UAV hardware. For it to simulate flight or driving, a simulation environment needs to be attached to it to provide information such as simulated world and sensor data and to resolve the dynamics of the vehicle. The simulation environment used in this project is Gazebo [50], which integrates seamlessly with the PX4 SITL simulator. This configuration is able to simulate up to 10 (and possibly more) UAVs, UGVs or any other type of vehicle for which models exist. The simulations communicate with the system through the UDP (user datagram protocol) protocol, and for the purposes of the conflict management framework they behave exactly like real hardware would. Simulated and real vehicles can be freely combined.

The X500 is a UAV development kit, from which a complete UAV can be built that uses the Pixhawk 4 flight controller, running the PX4 autopilot software. This is a very common configuration and probably provides the best solution in terms of achievable functions and ease of expandability. The other hardware, "PX4 in a box" also uses a Pixhawk hardware as the flight controller; however, it only includes the most essential sensors, so that the unit is able to connect to the system and provide telemetry data. There is no frame or propulsion unit, so it is not flight capable; rather, it is built into a box, which can be mounted on vehicles or even carried by hand, enabling integration into the system for arbitrary users. After boot, the box's controller needs to be "tricked" into believing it is flying or driving, which is achieved using the GCS software. Figure 4 shows this hardware as used during system tests.

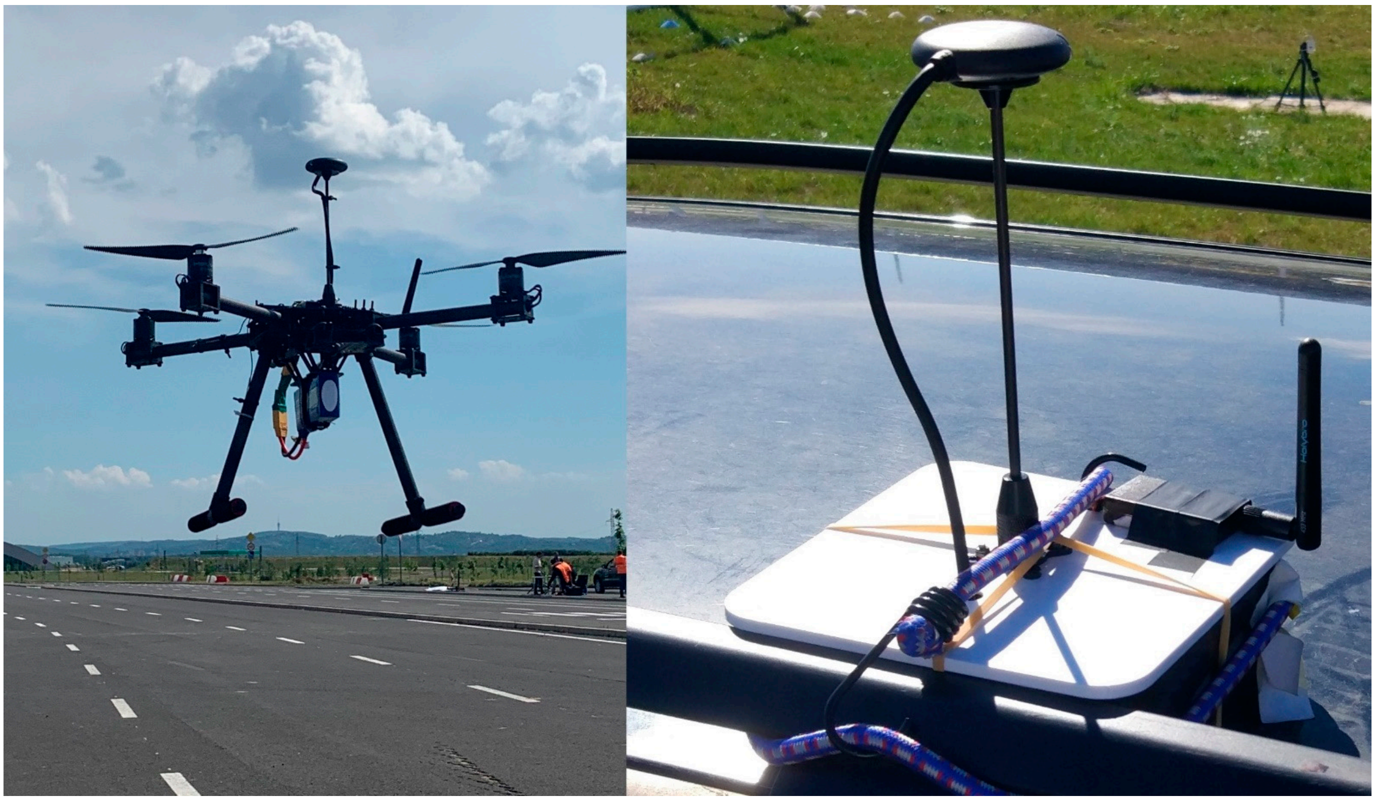
Figure 4. PX4-based real hardware: X500 (**left**) "PX4 in a box" (**right**).

The PX4-based users connect to the system via telemetry radios (the particular type of hardware is called SiK radio), sending telemetry data and commands as MAVLink messages via serial communication. This solution is one of the most widespread and available for open-source drones today. They operate on nominal 433 MHz using a frequency-hopping method to enable a number of these units to work simultaneously without interference. One end of the unit is connected to the vehicle and the other to the PC running the conflict management framework. These radios are primarily designed to work in pairs, and while multiple units could be connected for multi-point comms, in the authors' experience, it does not work suitably for this purpose. Considering this, during the operation of a production version of the conflict management system, these radios will be plugged in to the computers of the users who are operating the drones and not to the central conflict management computer. These telemetry radios are used by the operators to monitor the status of the UAV and manage missions. Routing and forwarding the communication to the system via UDP or TCP (transmission control protocol) can relatively easily solve this problem. For other type of radio systems, different interfaces need to be implemented, and this is part of the longer-term development plans.

3.1.3. Conflict Detection Methods

For the initial prototype of the system two conflict detection methods are implemented, both of the deterministic class, and they detect conflicts between pairs of users, initially only in two dimensions:

- Pairwise waypoint-based static area detection;
- Pairwise dynamic projected area detection.

In the first method, it is assumed that both users provide their mission definition, and as such the trajectories they are going to take are determined in advance. An "s" ratio is introduced, which represents the ratio of the mission trajectory already completed by the user. This "s" ratio in theory simplifies the detection as the ratios where conflicts could potentially arise during the mission of the users can be predicted in advance, as soon as a user registers (connects) to the system. It is then only important to know what the current "s" ratios of the vehicles are (presuming the mission definition also includes velocity targets) and determine if a possible conflict is active or not based on the ratios. Figure 5 shows an example visualisation of the detection of the conflict areas using this algorithm. A uniform safety buffer of 5 m was used on both sides of the defined mission trajectory (middle black and grey lines).

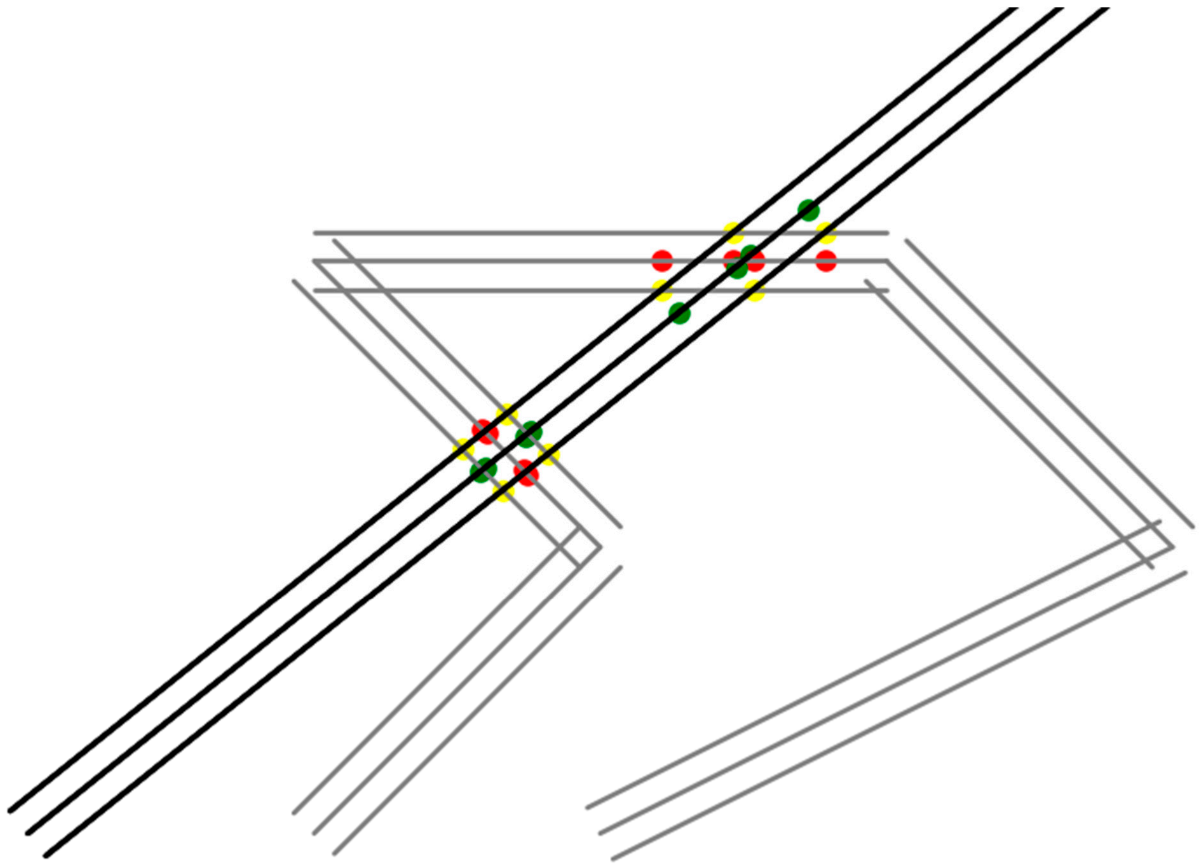

Figure 5. Pairwise waypoint-based static area detection method example; trajectories with safety offset (black and grey), determined conflict areas (yellow), "s" ratios along trajectory when conflict can be active (green for black trajectory and orange and red for grey trajectory).

While the algorithm in theory simplifies the detection and frees up resources from checking for conflicts where they simply cannot arise, the practical implementation showed that there are issues with the robust and accurate determination of the vehicles' "s" ratio. Most of the autopilot software solutions give some form of feedback regarding what the vehicle is doing; however, in the experience of the author, the current mission waypoint information is not always robust enough. Predicting which waypoint is the target without receiving it via telemetry also fails when the trajectories become more complex, such as self-intersecting trajectories, repeating waypoints, etc. Furthermore, any deviation from the flight path, including auto-land or return-to-home emergency functions, means that during these deviatory manoeuvres the system does not provide conflict management capabilities. Based on these results we decided not to pursue this form of conflict detection at the moment but to revisit it at a later stage of the research project to find potential solutions to the issues.

The pairwise dynamic projected area detection method works by calculating the predicted position of the user based on the current position and velocity, received as telemetry data. The predictions are updated at every cycle of the conflict management algorithm. The prediction time and a safety buffer are set for each user in each conflict pair. This allows the system to customise each conflict; for example, for users that operate with higher uncertainty a larger or differently shaped buffer area could be used, and slower responding users could have longer prediction times. Figure 6 shows an example of this methodology during testing in the first version of the GUI. Note the axes in Figure 6 are based on the UTM33N (EPSG:32633) coordinate system, and as such they include the full coordinate values to the points. Coordinate systems need to be converted often when using the framework system, as for example, Smart records coordinates in UTM33, but DJI and PX4 systems report telemetry in WGS84. In the first iteration it decided to display vehicles in UTM33N.

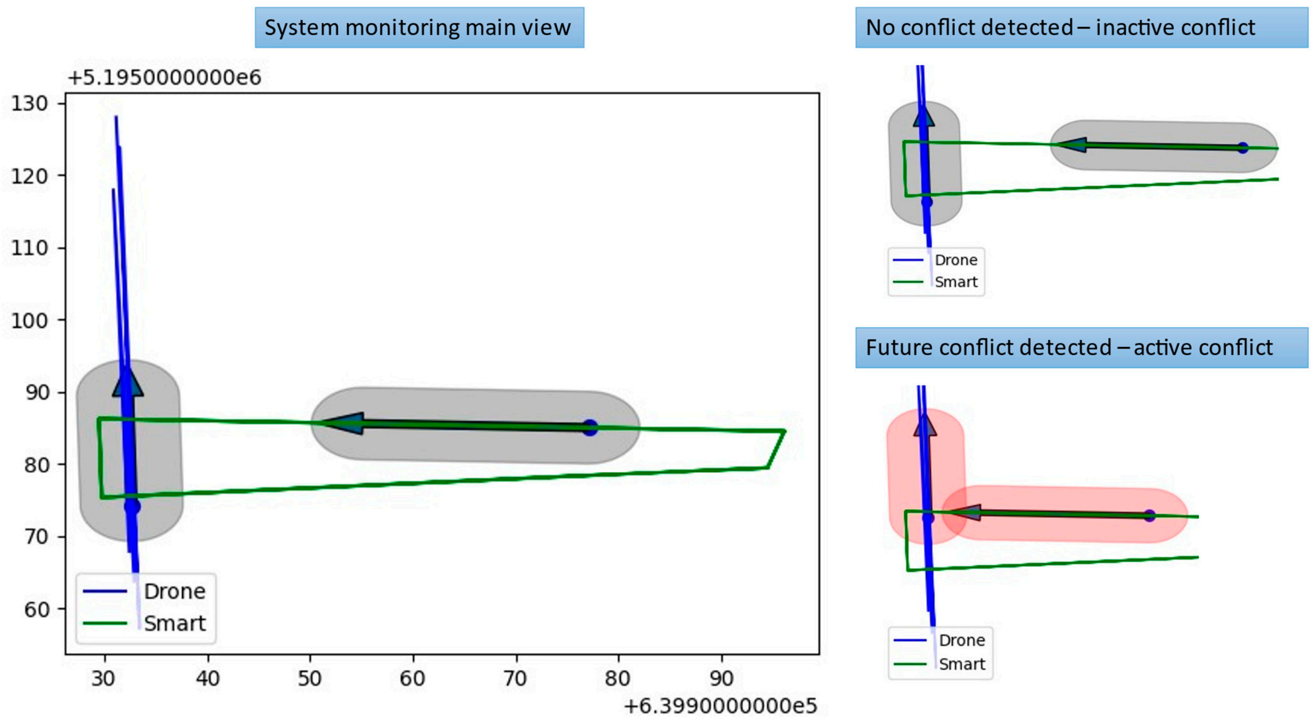

Figure 6. First version GUI display of pairwise dynamic projected area detection method; lines show user's mission (if available), points show the current position and arrows show predicted position. The grey (inactive) and red (active) areas show the conflict area with safety buffer applied).

Using the current and the predicted position and the safety buffer, a polygon is generated using the Shapely package of python. Shapely then can also be used to efficiently calculate intersections between the polygons of the two users. If the intersection area exists, then the conflict is set to be active, and it needs to be resolved. The method is very robust and quick, and as such it was set as the default algorithm for future development activities. An important aspect is the tuning of the prediction time and safety buffer parameters, which was performed using SITL simulations to yield an acceptable performance for the prototype system.

The system is intended to be used to develop and evaluate further methodologies, which will be modularly added in future steps of the development process.

3.1.4. Conflict Resolution Methods

Once the conflict detection method has established that a given conflict is active, then the allocated resolver method must provide a resolution for it. Detectors and resolvers are designed so they can be mixed within the framework and different combinations tested.

The baseline resolver that has been implemented is the stop-and-go resolution method. In this method one from the pair of vehicles in a conflict is stopped by the system, while the other is left to carry on with its mission. As one user can be involved in many conflicts, if a user receives a stop command from any of its conflicts, it must stop. Only those users who receive "carry on" commands (or rather, do not receive stop commands from any of its conflicts) can carry on with their mission.

To prevent impossible situations and deadlocks in the system, a priority assignment algorithm is created. The priority assignment considers first how a user is participating in the system. If the user is cooperative and it can receive and act on control commands from the system, it is assigned lower priority than a non-cooperative user. For each conflict pair, the user with the highest priority is given the carry-on command, and the lower priority user is sent the stop command. In the case of two non-cooperative users coming into conflict, no commands are sent, as the system has no means of affecting these users. In this case, it is up to the non-cooperative users to resolve the conflict to the best of their abilities. Non-cooperative users need to be dealt with at a different level, for example, by introducing policies that define mandatory standard protocols for vehicles, so that in theory they all become cooperative users. In the case where two cooperative users come into conflict, they will be allocated priority based on a set of rules. In the first implementation, the user's ID (unique for each user assigned when it registers to the system) is used to allocate priority, but this method can be used to give priority to less manoeuvrable or sensitive users, etc. It just needs to ensure that priority values are always absolute, and no deadlocks arise in multiple conflict pairs, in which all vehicles are given the stop command, and they all wait for each other forever.

The stop-and-go algorithm, despite its simplicity, works relatively well; however, it does not work for some cases, including coming into conflict head on, as one user would stop right on the trajectory of the second, possibly preventing its safe passage. For the second iteration of the system an improved algorithm is used, in which, in the case of head-on conflict, an evasive command is given to the lower priority vehicle before it is sent the stop command, so it moves out of the way. The robustness of this algorithm still needs to be improved, especially to make sure the lower priority vehicle is commanded to a position that is safe; this can be difficult to guarantee when multiple conflicts are active or the environment is confined.

When multiple vehicles are in conflict, how the multiple conflict is treated depends on the assigned conflict detection and resolution algorithms. Essentially, two types of algorithms can be defined, one which works between a pair of vehicles and one which resolves the detected conflict for an arbitrary number of vehicles. Obviously, when the pairwise algorithm is used, multiple algorithms are running parallel at the same time. When treating a very large number of vehicles, the management of the computational requirements needs to be considered. The number of conflict algorithms to be run in parallel scales as the factorial of the number of vehicles inolved, when pairwise treatment is chosen. The utilised stop-and-go algorithm for both versions of the management framework relies on pairwise detection and resolution, and due to the relatively low number of vehicles, computing capacity is of no concern.

Figure 7 shows the resolution process for a multi-vehicle conflict using the pairwise stop-and-go management algorithm. While all algorithms being developed have the capability to deal with multi-vehicle conflicts, as seen in the figure, the key focus at this stage of the research is the development of robust methodologies and testing with one pair of vehicles in conflict, as it is a lot clearer, easier and safer to track and control how a single pair of vehicles are behaving, especially during real hardware tests. In the following, discussion will be limited to a single pair of vehicles in conflict for this reason.

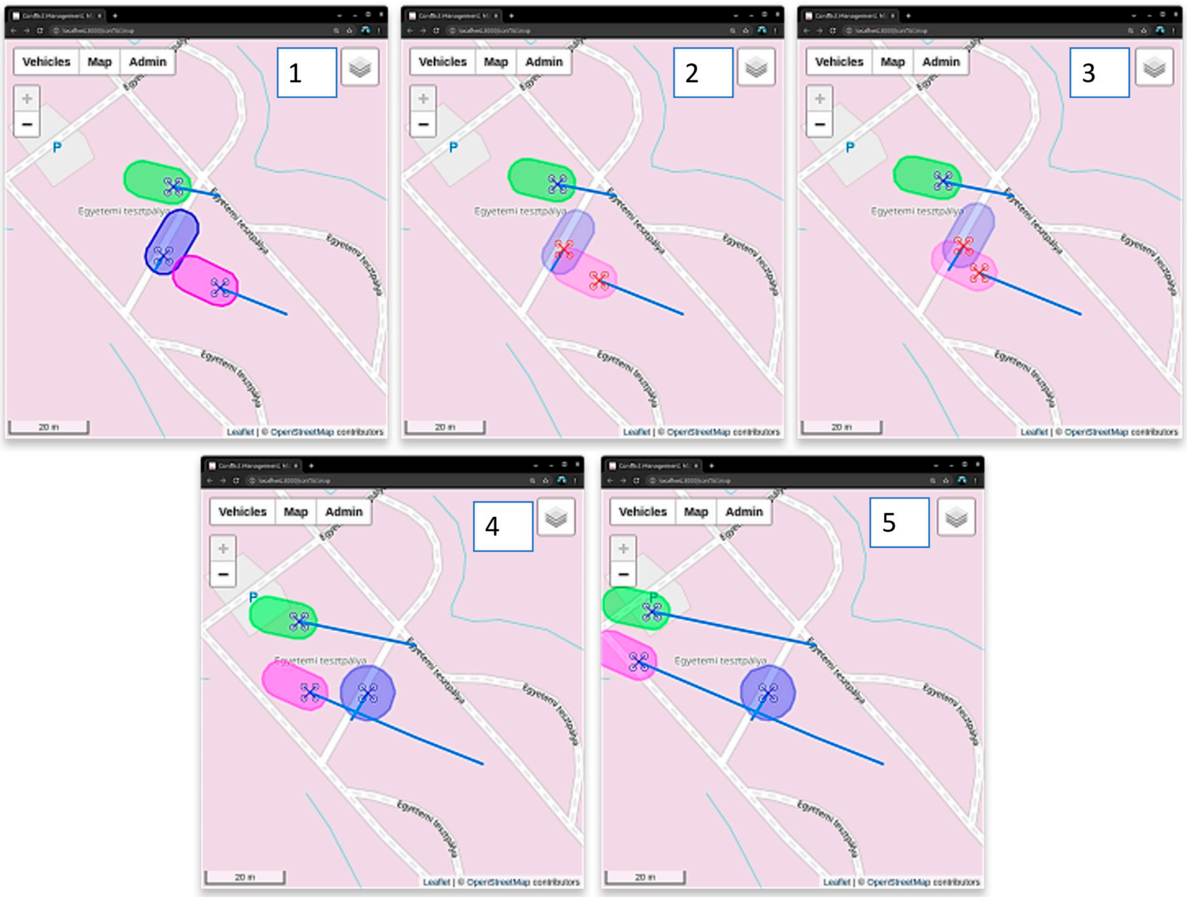

Figure 7. Multi-vehicle conflict resolution **1**,**2**: Conflict is detected between blue and purple, 3rd vehicle (green) unaffected; **3**: Lower priority (blue) vehicle is given command to stop and moves out of impact path; **4**,**5**: Conflict is resolved, vehicles not stopped carry on, commanded (blue) continues mission after safety delay. Non-English text are road section designations.

It is also worth mentioning that as the system does not deal with low-level flight control, as it only sends commands to the users. During testing it was observed that for some flight controllers, sending a pause or stop mission command would result in the UAV first stopping, then reversing to the position where it received the stop command. While it is debatable whether this is a feature or issue, it can be overcome, but the more important conclusion is that extensive testing is required for each type of user that is integrated into the system to understand how to best control them.

3.1.5. Lessons Learnt from Testing the First Version

The first version of the system was tested using primarily SITL simulators, but a successful real hardware test was also performed to validate the concept. During these tests some shortcomings were identified, which are presented and briefly discussed here.

Probably the greatest issue identified was the fragility of the software implementation. While the conflicts were managed safely when the system operated as intended, there were too many occasions when the operation of the system stopped. The primary culprit was the communication system. It was too easy for the telemetry radios to drop the connection between a user and the system, even if just momentarily. Since the connection to the users was established during initialization, this meant that the whole system had to be restarted to reconnect. The implementation of the MAVLink connection used (MAVSDK-python) also seemed to be a fragile type of link, as when the connection failed it often took manual intervention and effort to reset everything and initiate the connection again. This is clearly not acceptable for an automated system. Because the tools used in this case are open source, it would be possible to develop the communication link; however, that would go against the principles of the system. It was decided that a solution is required where instead of fixing the link's fragility, the system would manage the connection to make sure it can always recover after a failure.

Improving the robustness of the system's other aspects was also a priority. One of the biggest issues was the lack of persistent storage of data. Whenever the system stopped for any reason, the only way to fix it was essentially to reset it and start again. This is particularly problematic for the missions, as the missions are also started during the initialization step. This means that if the users were already executing their missions when it was restarted, they would first return to the first waypoint to start the mission from the beginning. This has undesirable consequences, as most controllers send the vehicle on the shortest possible direct route when the return command is received, irrespective of the conflicts that could be caused. This also makes testing difficult. When investigating specific conflict situations, the return activity throws off the timing required for the situation to arise, because it is not known from which point of its trajectory the vehicle will start its return when the system is restarted. While there was an attempt to mitigate these issues by not always restarting the missions when the system restarted, it was decided that a robust solution would be required rather than a workaround.

Another issue identified was the restriction of user interaction possibilities. When the system connects to MAVLink capable users it blocks the ports necessary to run GCS software. As such, the user loses its ability to monitor and control the vehicle's mission. This is a most undesirable outcome, as the system's role is only to manage conflicts; it must not take away the mission management possibility (and responsibility) from the users. While this could potentially be fixed by implementing GCS functionalities in the system and forcing users to use the system for their mission management, this also goes against the principle of relying on COTS components.

Among the issues there are also minor elements, such as the low quality of GUI implementation. As can be seen in Figure 6, the first iteration uses a Matplotlib plot to display the system status and conflicts. While it would be possible to develop this further, it would inherently be restrictive, as it can only be displayed on the computer running the system. Additionally, developing new GUIs when there are other possibilities that could be implemented and would probably work well is a waste of resources and goes against the LEAN principles. A final issue that also arises partly from the GUI implementation is that users cannot be added or removed from the system during runtime—the system needs to be stopped, and new users added in the code. While a production system would need to have automatic user management capabilities, for the prototype it would be useful if the system operator (supervisor) could add or remove users as required.

To summarise, the following shortcomings had to be addressed in the second version:

- Fragile connections;
- Difficult timing of scenarios for testing due to resetting;
- Persistent data storage;
- GCS use is prohibited;
- Low-quality GUI;
- Local monitoring capability only;
- Dynamic user management.

3.2. *Second Version of Software Implementation*

To address the shortcomings identified, a significant redesign of the system was required. This section discusses the second implementation of the conflict management system, focusing particularly on the features that solve the problems identified.

3.2.1. System Architecture

The principal change in version 2 was a change in the system architecture from a locally running code to a networked web-based application. The framework used was the python-based Django web framework (version 3.2). Django is a very popular framework for web application development with powerful capabilities and a large database of expansion modules. The default implementation of Django is a synchronous service, which means that when long-running tasks are initiated, the service temporarily becomes unresponsive until the task is resolved. This is not desirable for the conflict management, as the various activities need to be run parallel (asynchronously). In order to achieve this the Celery (version 5.1.2)-distributed task queue was integrated into the system. The application also relies on a PostgreSQL relational database server for persistent data storage and the RabbitMQ message passing interface for managing tasks. A high-level overview of the system architecture is shown in Figure 8, and the significant changes compared to the first version are discussed in the following section.

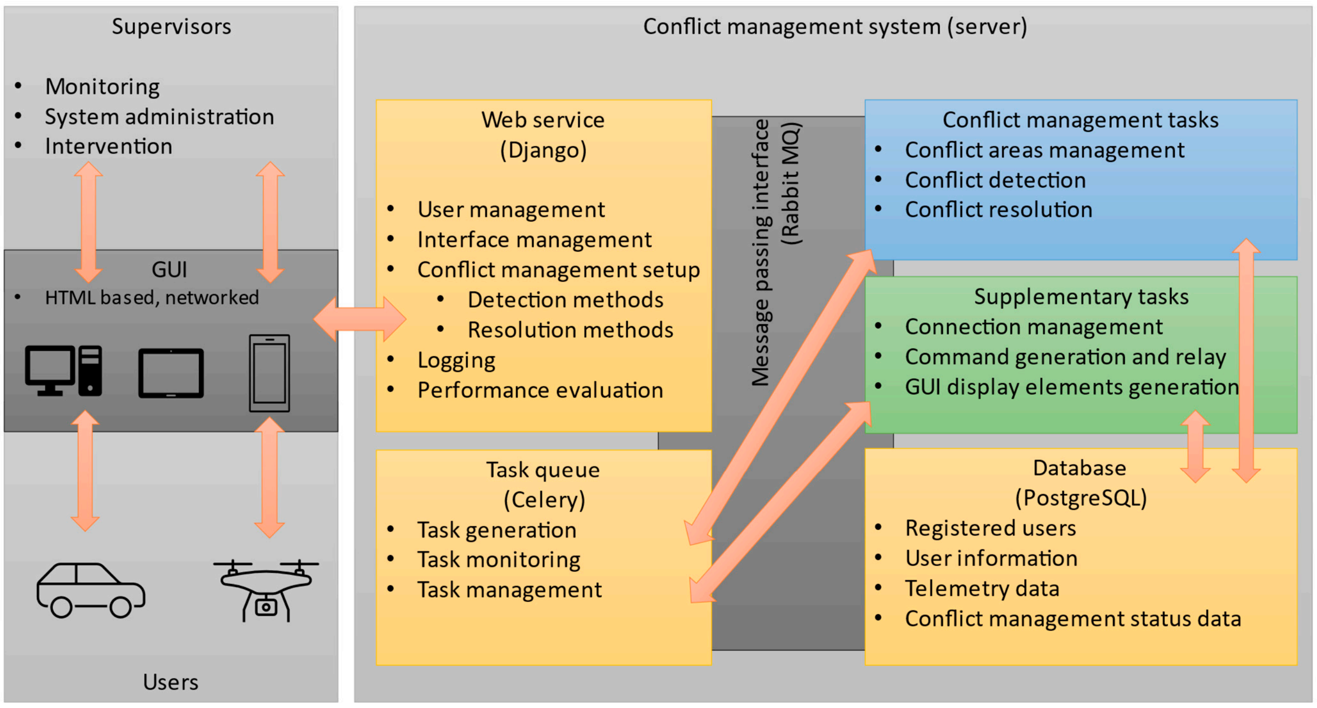

Figure 8. Conflict management system architecture—second version.

3.2.2. Communication Solutions

The introduction of the new architecture enables the implementation of robust communication solutions. In the current setup, each user is connected to the system by generating an individual connection manager task that is responsible for monitoring the health of the specific connection. The manager task is run periodically, and when the connection drops, it reinitiates the connection task. To facilitate this all tasks are designed to deliberately fail if they encounter non-desirable situations, implementing a fail-safe layer in the system. This is important, as tasks that are seemingly running but are stuck in a non-operative status can provide misleading information, which can result in dangerous situation during conflict management.

3.2.3. Data Persistence

The second version of the system uses a relational database to store data in a persistent and accessible manner. This is beneficial in two ways. First, it provides the system with an enhanced restart capability, as the relevant data are immediately available from the database even after a failure. Additionally, this approach enables the robust and parallel (asynchronous) handling of user data collection and the conflict detection and resolution tasks. Unlike the first version, where for each iteration of the conflict management loop, the system has to wait for each user to provide updated data, the data collection and usage run as separate tasks. A typical database system such as the one used is capable of handling in the order of 100,000 transactions per second with minimal tuning. Considering that the typical frequency of telemetry data acquisition from users is in the order of 20 Hz or less, the system should be capable of handling a large number of users without performance issues. In the case of a production system with a large number of users, the appropriate IT architecture planning (distributed databases, optimised transactions, etc.) solutions must be implemented. From the point of view of this research, these steps are not relevant, as it is known that solutions exist for this purpose, and it was not investigated further.

The database also has useful built-in utility functions, for example, timestamping the last modification of a table in the db. This is used, for example, to check the "freshness" of the collected telemetry data. From the system's standpoint, whether a user's connection is alive or not can be determined from the time the last telemetry data were written to the database. In the current system 3 s is set as the threshold. If the data are not "fresh", then the system treats the user as not connected, i.e., non-cooperative, until the connection manager reconnects the user. Figure 9 shows the second iteration of the GUI, displaying a user with timed-out telemetry messages and showing the indication of an active conflict zone.

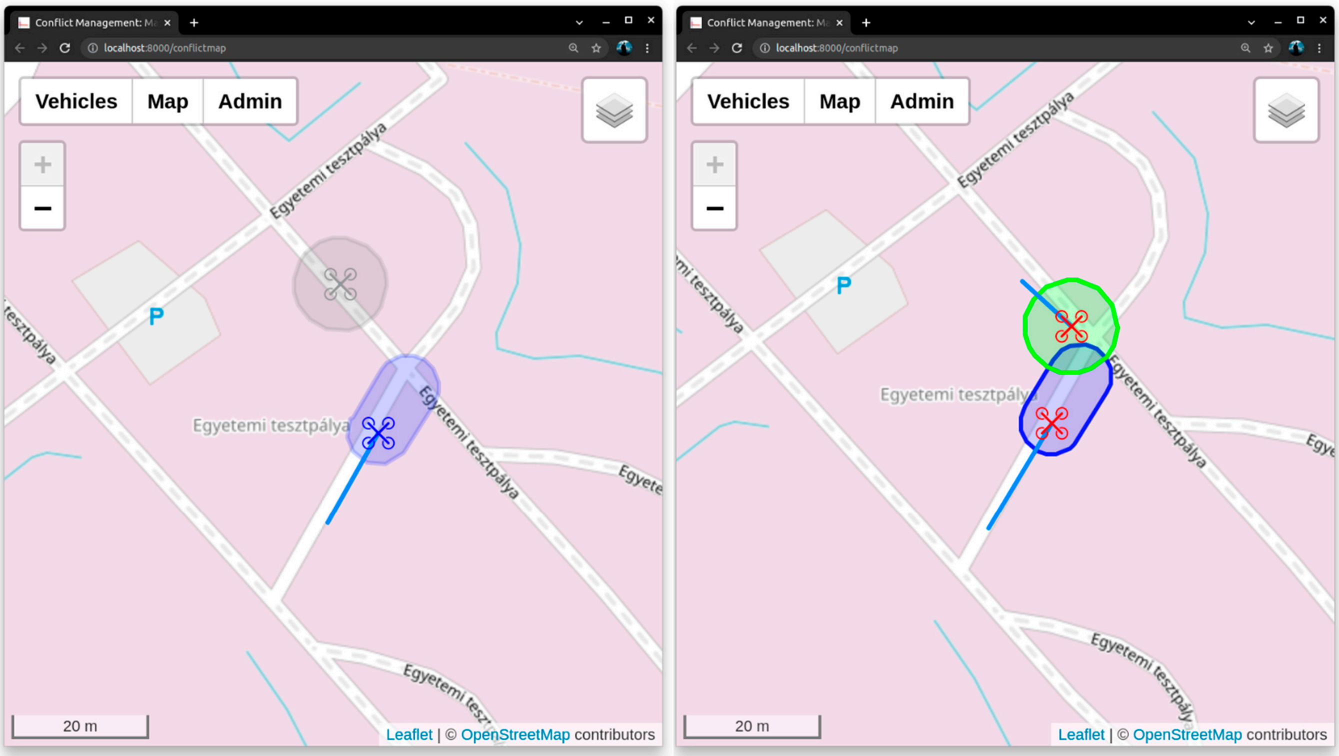

Figure 9. Version 2 GUI: indication of timed out telemetry (**left**), conflict area marked as active (**right**). Non-English text are road and area designations.

The colour coding of the current GUI is the following:

- Each vehicle is allocated a unique RGB colour during vehicle registration. (Red is not preferred, however can be allocated—it is purely a cosmetic question.) This colour is used to represent the conflict area when the conflict is not active.
- Vehicle icons by default are coloured in blue, as it was decided that the vehicle icon colour should represent the status of the vehicle.
- Trace of vehicle paths are marked with blue polylines.
- Vehicles with timed-out telemetry (lost connection or vehicle malfunction) are seen at their last reported location and last conflict area with greyed out colour.
- Vehicles in conflict are shown with flashing, bold line contoured coloured areas and flashing icon alternating between red and blue colour.

3.2.4. COTS Solutions Integration

Since the system is to be added as an additional security layer, not one replacing operator control, changes have been made to the way the users are connected. The PX4-based units are still connected through the SiK telemetry radios; however, the system no longer uses the connection COM port directly. The MAVLink-router package is used to run a forwarding service for each user, which in effect creates a mirrored copy of the connection port using the UDP protocol. This enables the computer running the system to connect to a user and a GCS at the same time. Or it can just receive UDP communication from a completely different computer, which connects to the user and its own GCS via the telemetry radio, as would be the case for an operating user. This way, the interface for connecting a simulated vehicle or a real hardware becomes identical as well. The same strategy was developed for DJI drones (using Windows or Mobile SDK—Software Development Kit), and it is currently in the process of being implemented. Integrating DJI SDK and MAVLink-based vehicles accounts for a very large segment of the current UAV solutions. MAVLink is a viable option for UGVs as well, either connecting to the vehicle itself or deploying the "PX4 in a box" solution. The use of pure position data from an arbitrary vehicle architecture is also a valid approach; however, this would provide only one-way communication, and thus would result in a non-cooperative user.

3.2.5. Graphical User Interface and Human Interaction

The GUI for the second version was significantly upgraded. Instead of using a single computer-based application and graphics, the system GUI is now served by the Django framework. It is an HTML response which can be viewed by any compatible device running an internet explorer (Edge, Firefox, Chrome, Safari, etc.). The base of the GUI is a Leaflet map. Leaflet is an open source javascript-based mapping library. It is simple to integrate into an HTML (HyperText Markup Language) view, and it is interactive and handles Geojson inputs. Geojson is a standard format for encoding various geographical features, such as points, paths, areas, etc. Plotting the users' positions, the conflict areas with the safety buffers, geofencing areas and other GIS features naturally lends itself to Geojson representation. From the conflict management related data stored in the database, the Django framework generates the Geojson features to be displayed on the Leaflet map, which is served to an arbitrary number of users via the web service.

The Leaflet map is completely customisable, the tilesets (background) of the map can be selected from different providers and the items displayed can be placed in layers that can be made hidden by the user and other similar features. Interacting with a map item (click or touch) can reveal additional information on the specific item in the form of a popup window or a separate HTML view. While this high level of customisation is beneficial, we was decided to deliberately limit the functionalities of the GUI to the conflict management specifics. A typical GCS has many options for setting up the vehicle, the mission, emergency functions and similar. It is our intention to keep these features separate, so the individual operators have to rely on their GCS and only use the conflict management system for its core purpose. This is important for maintaining the concept of relying on

COTS solutions. However, integrating the conflict management system into existing GCS is an option that will be considered in the future.

3.2.6. Improved Stop-and-Go Resolution Algorithm

For the second iteration of the system, the stop-and-go algorithm was also improved based on the lessons learnt from the first iteration. This chapter describes the algorithm used, highlighting the improvement and the reasons behind it for the second iteration.

In the first iteration of the methodology, as soon as the conflict was detected between two users, the system issued a "stop mission" command. This command is fairly universally present in the implementation of various flight controllers and their SDK mechanisms; however, it presents some significant drawbacks. First of all, the functionality might not exist for all COTS systems. Second, the implementation might not be done in the same way for different systems. The implementation in the PX4 flight controller stack used extensively in the research project suffers from a drawback; when the command is issued, the vehicle decelerates, stops, then reverses to the point where the vehicle has acknowledged the stop command. This is an undesirable behaviour, as the stopping point cannot be set deterministically due to the unknown latency in the communications systems. Furthermore, because the vehicle passes through the commanded stopping point, there is no guarantee that an impact is avoided in a conflict.

Upon investigating the specifics of the controller algorithm, the reason for this behaviour was found. Figure 10 shows the controller algorithm used by the flight control software. It can be seen that the system relies on a single proportional constant to ensure the position hold capacity of the vehicle. It can also be seen that the output of the system is a velocity control signal, which is saturated (limited) to prevent excessive accelerations. While it would be possible to include a derivative (and if necessary, an integral) constant to prevent overrun and reversing in the system, one of the key principles of the conflict management system is to not modify the individual control algorithms of the vehicles.

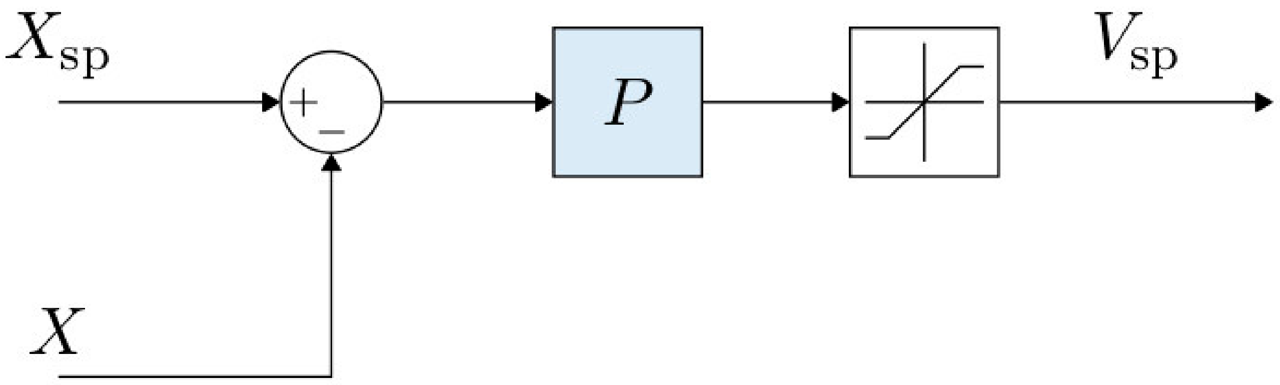

Figure 10. PX4 controller algorithm for position control (source: [51]).

The solution, which is adopted in the second iteration of the stop-and-go algorithm, is to use a "set position" command, instead of the "stop mission" command. When using this command, the vehicle dynamics are considered to issue a command where the accelerations can be kept below the saturation level of the controller. While this does not completely eliminate overswings while acquiring positions, in practice these overswings are reduced to the order of 10–20 cm, and as such they are no longer of concern.

Using set position commands also mitigates another big drawback of the stop-and-go algorithm, specifically, that it cannot handle head-on or shallow angle conflicts, such as if one vehicle is stopped in the path of the other one. Although it avoided immediate impact, this did not actually resolve the conflict. Using set positions allows the command to define the stopping point, so that the vehicle would stop outside the predicted conflict area of the other vehicle given the go signal. Due to the vehicle dynamics this stopping area needs to be chosen within a fairly narrow angle cone, below 30 degrees half angle measured from the velocity vector of the commanded vehicle. The quoted angle can be even more restrictive based on velocity, conflict prediction time and safety buffer size. In the second iteration, based on the test scenarios performed, to achieve acceptable behaviour, the safety buffer was set to a 4.5 m radius, and the conflict detection time was set to 4.0 s. The position target for stopping is derived based on 3.0 s of resolution time and the current vehicle velocity, which results in a distance that can be mapped within the cone defined by the dynamics. These specific settings do not necessarily provide an optimum behaviour, but rather they are settings that were chosen based on a trial-and-error process and seem to provide an acceptable qualitative performance for the algorithm in the test scenarios. For quantitative evaluation, please refer to the discussion section of this paper.

A final modification of the stop-and-go methodology involved the conflict detection algorithm. When the vehicle is commanded to stop, its conflict area will gradually reduce as its speed drops (refer to Figure 6, Figure 7 or Figure 9 for conflict area visualizations). In many cases during the tests in the first iteration of the system, this drop resulted in a situation where a stopped vehicle's conflict zone no longer intersected the conflict zone of the other. In this case, the algorithm assumes that the conflict is resolved, and the stopped vehicle is given the go signal, effectively sending it in front of the other vehicle, creating a significant impact risk. In the second iteration this issue is resolved by increasing the safety buffer of stopped or slow-moving (below 0.25 m/s velocity) vehicles to 1.5 times the defined conflict buffer distances. This allows the stopped or almost stopped vehicles to remain in conflict and stopped.

4. Discussion

This chapter summarises and discusses the insights gained from the testing of the prototype system and proposes further development directions. There has been extensive testing involving the SITL simulator, and it was confirmed that the robustness of the system and the communications layer has been increased significantly. Real hardware tests also showed that the second version provides a superior performance. The two performance criteria used in the case of prototype testing were qualitative evaluations of system functionality (which is effectively safety for the vehicles) and the robustness of the system. Regarding robustness, the first implemented method required resetting the system and users when non-desirable behaviour occurred; this involves returning the vehicles to the starting positions, restarting the management algorithm and disconnecting and reconnecting the communication systems (telemetry radios) of the vehicles. This can up to 1–5 min per reset. Note that the average endurance for drones is in the order of 30 min, so even a few resets take up significant portion of their available flight time. Recalling drones to replace batteries during the tests or demonstration would add an undesirable amount of waiting time, and in extreme cases could prevent the successful completion of all scenarios during a test day. Due to the persistent data storage and differing conflict resolution algorithm, the second iteration does not require resetting; furthermore, in the case of communication loss, it automatically attempts to reconnect to the users. In terms of the other criteria, the system functionality, the first implemented version of the stop-and-go algorithm showed a number of mismanaged conflicts, e.g., the lower priority vehicle did not stop at a discovered conflict, or first stopped and continued its mission. Using the second algorithm eliminated these system errors, and in the scenarios tested, it provided appropriate resolutions for all conflicts. Quantitative evaluation of system safety, management efficiency (time lost due to conflict management) and robustness is being carried out in the current, follow-up stage of the research project.

The current state of the research project was also presented at a live demo conducted in the ZalaZone Proving Grounds in Hungary. The demo was aimed at demonstrating the core principles of the conflict management system. Unfortunately, during the demo, the allocated space was rather limited, and only a single intersection of roads was made available, so the scope of the demonstration was also limited. Refer to Figure 11 for photographs taken at the event. In the demo, the UAV has received a repeating mission to fly between the two sides of the intersection. The ground vehicle had the PX4-in-a-box unit mounted to the roof and was operated by a driver. A manned ground vehicle seemed like the appropriate choice for such limited space, as the intention was to demonstrate that the system can react to a wide variety of situations that can arise. Relying on a driver means that the driver can also improvise in the situations and demonstrate the safe workings of the system. The demo was successfully presented, and while the system's performance was excellent, some areas for further improvement were identified.

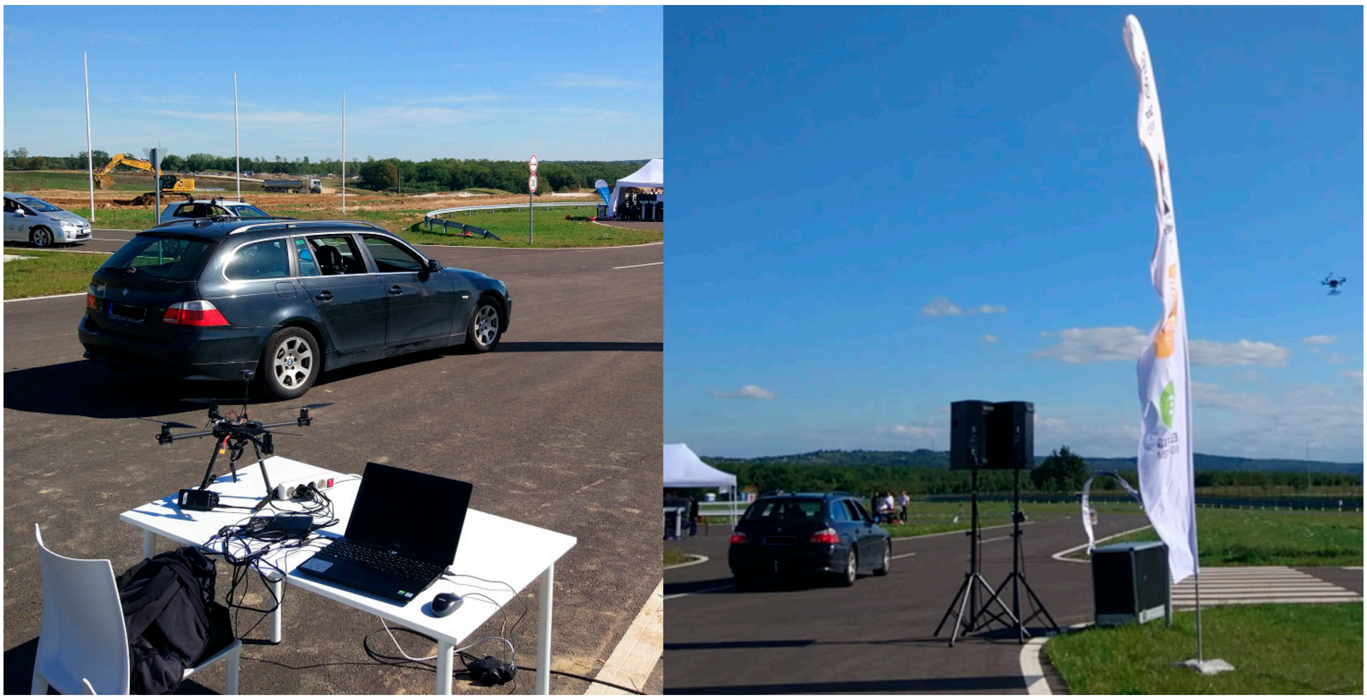
Figure 11. ZalaZone demo setup: ground control station and system users (**left**), system in operation (**right**).

Among the identified areas to improve was the aforementioned case, when head-on conflicts occur. In this case, the stopped vehicle needs to perform an evasive manoeuvre before stopping to ensure that separation is maintained between the users. To do that safely, in addition to the system users' positions, information about the environment is also required. As such the next important step is the integration of a GIS system.

Another direction is the closer integration of DJI products. The Windows SDK was successfully tested, and basic automatic operation was demonstrated. There are significant limitations in this SDK version, and based on the lack of updates it seems that it might be on the road to obsolescence. As such the Mobile SDK (Android and iOS) is under consideration. Mobile development can also unlock new opportunities, as it would enable smartphones to join the system as users. However, mobile development is a separate niche of programming, and the costs and benefits need to be weighed before effort is invested in building up this technology skillset in the research group.

As the framework's performance is acceptable, the next primary direction of the research group is to develop the conflict management algorithms. For the conflict detection algorithm, the most important thing is to include information from vehicle trajectories (mission plans), if available. Knowing the trajectories can potentially help to resolve some detected conflict situations more efficiently.

Regarding efficiency, another key development direction is a method to objectively evaluate the system's safety, performance and efficiency measures. While intuitively, a simple resolution algorithm such as stop-and-go seems to be less efficient than more advanced ones, this might be based on the inconvenience experienced as a human driver when the vehicle needs to be stopped or the route changed. For an autonomous system, however, convenience is not a factor that matters. Unless specific criteria are defined, it is impossible to compare methods objectively. Once the measurement method is developed, SITL simulations can be used to objectively compare detection and resolution methods. Furthermore, optimisation tools can be used to determine the optimum parameter settings for a specific type of algorithm.

In the follow-up studies since the completion of the first stage of the project the development of objective quantitative performance criteria is one of the key points of research. The following list highlights the criteria that are being used in the development for the system:

- Safety measures:
  - Number of users in the system: this is used to calculate the per-user performance measures to compare management of areas with varying traffic intensity;
  - Number of conflicts detected: the number of times the conflict detection methods turn the conflict areas active;
  - Number of resolution commands issued;
  - Number of mismanaged conflicts;
  - Number of impacts between users;
  - Number of near impacts between users.
- Performance measures:
  - Time and distance travelled by individual users without other users or conflicts;
  - Time and distance travelled by individual users while other users are present, and the conflict is being managed by the system;
  - Highest level of accelerations and decelerations commanded by the system;
  - Total change in altitude commanded by the system (only applicable to UAVs);
  - Time and distance spent off-road or parked (UGV);
  - Time spent landed, if commanded (UAV).

When evaluating the performance criteria, the following approach is being developed:

1. Simulate individual users, alone in the conflict management area, to evaluate how they would perform their mission without interactions from other users or the conflict management system.
2. Simulate (or when sufficient confidence in the system is achieved, live test) the conflict area with all users involved and log the individual positions, velocities, accelerations and commands received.
3. Post-process the logged information to evaluate the performance measures listed above.

The three-stage process described enables the objective evaluation of the performance measures, as it essentially compares the theoretical possible unrestricted operation of all users to the actual performance they can achieve while multiple users are around, and the conflict is managed. The best management solutions provide the highest level of safety, so no impacts or near impacts and detect all conflicts, while at the same time being the least intrusive, so issuing a minimum number of resolution commands and imposing minimum additional time and distance (detours). Quantitatively evaluating these measures enables the objective comparison and development of different conflict management methods and strategies, which is the ultimate aim of the research framework presented here.

5. Conclusions

In conclusion, in this research a system framework was developed and its functionality successfully demonstrated. The framework implements short- and mid-term tactical conflict management for a wide user base, including unmanned aerial and ground vehicles. The framework has gone through two iterations, the second of which significantly improved key issues that were identified during development and testing of the first. The final, second version of the framework achieves a technology and solution independent implementation of the stop-and-go conflict management algorithm. This system is capable of handling a wide range of unmanned and aerial ground vehicles without introducing any modifications to the communications or control systems of any particular vehicle. The framework is implemented as a modular suite of software algorithms, with simulation software integration. The framework is accessible as a web service, where an arbitrary number of users and supervisors can connect to the system using standard web browsers from various devices to monitor activities and perform administrative tasks. The system is designed at the moment to handle up to 250 users (vehicles). The web service framework relies natively on database persistent information storage, which improves the robustness and recovery characteristics of the system, including reconnecting communication systems when the connection is interrupted. The stop-and-go algorithm was also improved to consider system dynamics and head-on conflict cases.

The framework's performance is considered acceptable, and future research focus is based on conflict management methods development and evaluation.

Author Contributions: Conceptualization, D.R. and D.S.; methodology, D.S.; software, D.S.; validation, D.R. and D.S.; formal analysis, D.S.; investigation, D.R. and D.S.; resources, D.R. and D.S.; data curation, D.S.; writing—original draft preparation, D.S.; writing—review and editing, D.S.; visualization, D.S.; supervision, D.S.; project administration, D.R. and D.S.; funding acquisition, D.R. and D.S. All authors have read and agreed to the published version of the manuscript.

Funding: The research reported in this paper and carried out at the Budapest University Technology and Economics was supported by the National Research Development and Innovation Fund (TKP2020 National Challenges Subprogram, Grant No. BME-NC) based on the charter of bolster issued by the National Research Development and Innovation Office under the auspices of the Ministry for Innovation and Technology. In addition the research was supported by the National Research, Development and Innovation Office through the project "National Laboratory for Autonomous Systems" under Grant NKFIH-869/2020. Funder NKFIH Grant Nos.: TKP2020 BME-NC and NKFIH-869/2020.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors would like to acknowledge Gábor Jandó and Károly Veres for their contribution to the extensive testing and operation of the system during demo events.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Poponak, N.; Porat, M.; Hallam, C.; Jankowski, S.; Samuelson, A.; Nannizzi, M. Drones Flying into the Mainstream. 2016. Available online: https://www.goldmansachs.com/insights/pages/drones-flying-into-the-mainstream.html (accessed on 22 October 2021).
2. MarketsandMarkets. Unmanned Aerial Vehicle (UAV) Market by Point of Sale, Systems, Platform (Civil & Commercial, and Defense & Governement), Function, End Use, Application, Type, Mode of Operation, MTOW, Range, and Region—Global Forecast to 2026. In *Markets and Markets*; MarketsandMarkets Research Private Ltd.: Hadapsar, India, 2021. Available online: https://www.marketsandmarkets.com/Market-Reports/unmanned-aerial-vehicles-uav-market-662.html?gclid=CjwKCAiA78aNBhAlEiwA7B76p3r10jhv4HYaPU5K6iYWi-9Uq1S_SlE_W5bEvNb2U6_unbt5QHiLZhoC_FoQAvD_BwE (accessed on 22 October 2021).
3. *Global Drone Market Report 2021–2026*; Research and Markets: Dublin, Ireland, 2021.
4. *European Aviation in 2040, Challenges of Growth*; EUROCONTROL: Brussels, Belgium, 2018. Available online: https://www.eurocontrol.int/sites/default/files/content/documents/official-documents/reports/challenges-of-growth-2018.pdf (accessed on 22 October 2021).
5. European Automobile Manufacturers Association. *Vehicles in Use Europe 2019*; ACEA Report: Brussels, Belgium, 2019.
6. Litman, A.T. Autonomous Vehicle Implementation Predictions, Implications for Transport Planning. 2021. Available online: https://www.vtpi.org/avip.pdf (accessed on 22 October 2021).
7. On-Road Automated Driving (ORAD) committee, Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. *SAE Int.* **2021**. [CrossRef]
8. *Unmanned Aircraft Systems Traffic Management (UTM)—A Common Framework with Core Principles for Global Harmonization*, 3rd ed.; International Civil Aviation Organization: Montreal, QC, Canada, 2020.

9. Bradford, S.; Kopardekar, P. *Uncrewed Aircraft Systems (UAS) Traffic Management (UTM), UTM Pilot Program (UPP), UPP Phase 2 Final Report, Version 1.0*; Federal Aviation Administration and National Aeronautics and Space Administration Joint Publication: City, WA, USA, 29 July 2021. Available online: https://www.faa.gov/uas/research_development/traffic_management/utm_pilot_program/media/UTM_Pilot_Program_Phase_2_Progress_Report.pdf (accessed on 22 October 2021).
10. ASTM F3411-19. *Standard Specification for Remote ID and Tracking*; ASTM International: Conshohocken, PA, USA, 2019. [CrossRef]
11. *Initial View on Principles for the U-Space Architecture, European Union*; EUROCONTROL: Brussels, Belgium, 2019.
12. SESAR, GOF U-Space, European Union, SESAR JU GOF U-Space Project: Final Demo with Piloted Air Taxi Flight Successfully Completed, Helsinki. 2019. Available online: https://www.sesarju.eu/news/sesar-gulf-finland-u-space-project-first-demos-successfully-completed (accessed on 22 October 2021).
13. Federal Office of Civil Aviation. Swiss U-Space ConOps, U-Space Program Management, Reference: FOCA muo/042.2-00002/00001/00005/00021/00003. 2019. Available online: https://susi.swiss/wp-content/uploads/2020/04/Swiss-U-Space-ConOps-v.1.1.pdf (accessed on 22 October 2021).
14. Innovation Hub. *Beyond Visual Line of Sight in Non-Segregated Airspace, Fundamental Principles & Terminology*; UK Civil Aviation Authority: London, UK, 2020.
15. Eliot, L. Pairing Self-Driving Cars with Autonomous Drones for Faster Fast Food Delivery. 2019. Available online: https://www.forbes.com/sites/lanceeliot/2019/06/21/self-driving-cars-paired-with-autonomous-drones-for-faster-fast-food-delivery/ (accessed on 22 October 2021).
16. Nogar, S.M. Autonomous Landing of a UAV on a Moving Ground Vehicle in a GPS Denied Environment. In Proceedings of the 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Abu Dhabi, United Arab Emirates, 4–6 November 2020; pp. 77–83. [CrossRef]
17. Hament, B.; Oh, P. Unmanned aerial and ground vehicle (UAV-UGV) system prototype for civil infrastructure missions. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 12–14 January 2018; pp. 1–4. [CrossRef]
18. Cocchioni, F.; Pierfelice, V.; Benini, A.; Mancini, A.; Frontoni, E.; Zingaretti, P.; Ippoliti, G.; Longhi, S. Unmanned Ground and Aerial Vehicles in extended range indoor and outdoor missions. In Proceedings of the 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 27–30 May 2014; pp. 374–382. [CrossRef]
19. Waslander, S.L. Unmanned Aerial and Ground Vehicle Teams: Recent Work and Open Problems. In *Autonomous Control Systems and Vehicles, Intelligent Systems, Control and Automation: Science and Engineering*; Nonami, K., Kartidjo, M., Yoon, K.J., Budiyono, A., Eds.; Springer: Tokyo, Japan, 2013; Volume 65. [CrossRef]
20. Logan, M.J.; Bird, E.; Hernandez, L.; Menard, M. Operational Considerations of Small UAS in Urban Canyons. In Proceedings of the AIAA SciTech 2020 Forum, Orlando, FL, USA, 6–10 January 2020.
21. Low, K.H. Framework for urban Traffic Management of Unmanned Aircraft System (uTM-UAS), Drone Enable. In Proceedings of the ICAO Unmanned Aircraft Systems (UAS) Industry Symposium (UAS2017), Montreal, QC, Canada, 22 September 2017.
22. Krozel, J.; Mueller, T.; Hunter, G. Free Flight Conflict Detection and Resolution Analysis Paper AIAA-96-3763. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Diego, CA, USA, 29–31 July 1996.
23. Lijima, Y.; Hagiwara, H.; Kasai, H. Results of Collision Avoidance Maneuver Experiments Using a Knowledge-Based Autonomous Piloting System. *J. Navig.* **1991**, *44*, 194–204.
24. Coenen, F.P.; Smeaton, G.P.; Bole, A.G. Knowledge-Based Collision Avoidance. *J. Navig.* **1989**, *42*, 107–116. [CrossRef]
25. Radio Technical Committee on Aeronautics (RTCA), Minimum Performance Standards—Airborne Ground Proximity Warning Equipment, Document No. RTCA/DO-161A, WA, USA. 1976. Available online: https://my.rtca.org/NC__Product?id=a1B36000001IcnOEAS (accessed on 22 October 2021).
26. Bilimoria, K.D.; Sridhar, B.; Chatterji, G.B. Effects of Conflict Resolution Maneuvers and Traffic Density of Free Flight. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Diego, CA, USA, 29–30 July 1996.
27. Krozel, J.; Peters, M. Conflict Detection and Resolution for Free Flight. *Air Traffic Control. Q.* **1997**, *5*, 181–212. [CrossRef]
28. Havel, K.; Husarcik, J. A Theory of the Tactical Conflict Prediction of a Pair of Aircraft. *J. Navig.* **1989**, *42*, 417–429. [CrossRef]
29. Radio Technical Committee on Aeronautics (RTCA), Minimum Performance Specifications For TCAS Airborne Equipment, Document No. RTCA/DO-185, Washington. 1983. Available online: https://my.rtca.org/NC__Product?id=a1B36000001IcmZEAS (accessed on 22 October 2021).
30. Ford, R.L. The Conflict Resolution Process for TCAS II and Some Simulation Results. *J. Navig.* **1987**, *40*, 283–303. [CrossRef]
31. Tomlin, C.; Pappas, G.J.; Sastry, S. Conflict Resolution for Air Traffic Management: A Case Study in Multi-Agent Hybrid Systems. *IEEE Trans. Autom. Control.* **1998**, *43*, 509–521. [CrossRef]
32. Shepard, T.; Dean, T.; Powley, W.; Akl, Y. A Conflict Prediction Algorithm Using Intent Information. In Proceedings of the 36th Annual Air Traffic Control Association Conference Proceedings, Arlington, VA, USA, 29 September–3 October 1991.
33. Shewchun, M.; Feron, E. Linear Matrix Inequalities for Analysis of Free Flight Conflict Problems. In Proceedings of the IEEE Conference on Decision and Control, San Diego, CA, USA, 12 December 1997.
34. Ratcliffe, S. Automatic Conflict Detection Logic for Future Air Traffic Control. *J. Navig.* **1989**, *42*, 92–106. [CrossRef]
35. Paielli, R.A.; Erzberger, H. Conflict Probability Estimation for Free Flight. *J. Guid. Control. Dyn.* **1997**, *20*, 588–596. [CrossRef]
36. Carpenter, B.; Kuchar, J. Probability-Based Collision Alerting Logic for Closely-Spaced Parallel Approach, Paper AIAA-97-0222. In Proceedings of the 35th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 6–10 January 1997.

37. Bakker, G.J.; Blom, H.A.P. Air Traffic Collision Risk Modeling. In Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio, TX, USA, 15–17 December 1993; Volume 2.
38. Williams, P.R. Aircraft Collision Avoidance using Statistical Decision Theory. In *Sensors and Sensor Systems for Guidance and Navigation II*; International Society for Optics and Photonics: Bellingham, WA, USA, 1992.
39. Durand, N.; Alliot, J.; Chansou, O. Optimal Resolution of En Route Conflicts. *Air Traffic Control. Q.* **1995**, *3*, 139–161. [CrossRef]
40. Glover, W.; Lygeros, J.A. *Stochastic Hybrid Model for Air Traffic Control Simulation*; Springer: Berlin/Heidelberg, Germany, 2004. [CrossRef]
41. Visintini, A.L.; Glover, W.; Lygeros, J.; Maciejowski, J. Monte Carlo optimization for conflict resolution in air traffic control. *IEEE Trans. Intell. Transp. Syst.* **2006**, *7*, 470–482. [CrossRef]
42. Palme, R.; Siket, Z.; Gati, B.; Rohacs, J. Non-cooperative target classification, with use of their measured motion kinematics. In Proceedings of the 12th Mini Conference on Vehicle System Dynamics, Identification and Anomalies, Budapest, Hungary, 8–10 November 2010; Zobory, I., Ed.; BME Budapest: Budapest, Hungary, 2012; pp. 369–384, ISBN 978 963 313 058 2.
43. Szalay, Z.; Ficzere, D.; Tihanyi, V.; Magyar, F.; Soós, G.; Varga, P. 5G-Enabled Autonomous Driving Demonstration with a V2X Scenario-in-the-Loop Approach. *Sensors* **2020**, *20*, 7344. [CrossRef] [PubMed]
44. Szalay, Z. Next Generation X-in-the-Loop Validation Methodology for Automated Vehicle Systems. *IEEE Access* **2021**, *9*, 35616–35632. [CrossRef]
45. Tihanyi, V.; Rövid, A.; Remeli, V.; Vincze, Z.; Csonthó, M.; Pethő, Z.; Szalai, M.; Varga, B.; Khalil, A.; Szalay, Z. Towards Cooperative Perception Services for ITS: Digital Twin in the Automotive Edge Cloud. *Energies* **2021**, *14*, 5930. [CrossRef]
46. Szalay, Z.; Nyerges, Á.; Hamar, Z.; Hesz, M. Technical Specification Methodology for an Automotive Proving Ground Dedicated to Connected and Automated Vehicles. *Period. Polytech. Transp. Eng.* **2017**, *45*, 168–174. [CrossRef]
47. Szalay, Z.; Hamar, Z.; Nyerges, Á. Novel design concept for an automotive proving ground supporting multilevel CAV development. *Int. J. Veh. Des.* **2019**, *80*, 1–22. [CrossRef]
48. Totalcar, Akik Megtanítják a Smartot Önvezetni. Available online: https://totalcar.hu/magazin/2019/05/10/akik_megtanitjak_a_smartot_onvezetni/ (accessed on 22 October 2021).
49. Meier, L.; Honegger, D.; Pollefeys, M. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 6235–6240. [CrossRef]
50. Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2149–2154.
51. PX4 Documentation. Controller Diagrams, Multicopter Position Controller. Available online: https://docs.px4.io/v1.12/en/flight_stack/controller_diagrams.html (accessed on 22 October 2021).