Architecture for emergency control of autonomous UAV ensembles

Martin Schörner, Constantin Wanninger, Alwin Hoffmann, Oliver Kosak, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Schörner, Martin, Constantin Wanninger, Alwin Hoffmann, Oliver Kosak, and Wolfgang Reif. 2021. "Architecture for emergency control of autonomous UAV ensembles." In *3rd International Workshop on Robotic Software Engineering (RoSE'21), co-located with ICSE 2021, Virtual, Madrid, Spain, May 23 – 29, 2021*, 41–46. Piscataway, NJ: IEEE. https://doi.org/10.1109/RoSE52553.2021.00014.

Architecture for Emergency Control of Autonomous UAV Ensembles*

Martin Schörner[1], Constantin Wanninger[2], Alwin Hoffmann[3], Oliver Kosak[4], and Wolfgang Reif[5]
Institute for Software and Systems Engineering, University of Augsburg, Germany
Email: {schoerner[1], wanninger[2], hoffmann[3], kosak[4], reif[5]}@isse.de

***Abstract*— Applying unmanned aerial vehicles (UAV) has benefits for many different use-cases. Existing implementations of ground control stations (GCS) to manage UAVs in such scenarios already provide some support for the operation of multi-unit systems, i.e., ensembles. However, since they are usually designed for the operation of only one copter at once, this is often not sufficient to react quickly in dangerous situations, e.g., search and rescue scenarios. To address this problem, we propose an approach for easy observation and control of complete autonomous UAV ensembles: The Intention of our approach is to greatly reduce the number of personnel required for the operation of an UAV ensemble. Thereby, we generate the possibility for rapid intervention in potentially dangerous situations in order to prevent damage to the UAVs and the environment. In this paper, we present a software architecture for this safety-critical multi UAV ground control station including a fully implemented prototype which we also tested in a realistic environment.**

I. Introduction

Whether used for package delivery by Amazon and DHL, applied in search and rescue missions [1], for monitoring large areas [2], or as a light show in the night sky [3], mobile flying robots or unmanned aerial vehicles (UAV) currently become more important for research and industry. Due to their low cost, increasing availability, and flexibility, UAVs are well suited as participants in ensembles This is described in some of our earlier work [4], where UAVs cooperatively accomplish tasks in groups: When operating several independent devices in parallel instead of the operation of a single UAV, new requirements arise for the control and monitoring of the ensemble. In contrast to the ever increasing possibilities for different applications of UAV ensembles, approaches for their intuitive control unfortunately are still very limited. To fill this gap, we propose our approach of an Ensemble Control (ECO) ground station for UAV ensembles integrating possibilities for the intuitive identification, selection, and control of UAV groups. We intend ECO to be easily usable by a single operator that can achieve full control over the whole ensemble when necessary. This is relevant for all applications of UAV ensembles that are similar to the running example of our project COMBO that we also use to illustrate our findings in this article. Our goal in COMBO is to support rescue forces in major catastrophe scenarios, e.g., chemical accidents [4]. Tasks in this scenario are manifold, e.g., surveying an endangered area, searching and informing for civilians at risk, monitoring local weather conditions, or collectively transporting fragile materials [5] In all these tasks, being able to intuitively react to situations that call for user interaction is urgently needed. To achieve this, we illustrate our development of an appropriate user tool in the following. Therefore, we first identify the resulting challenges and highlight our objectives in Section II to be able to appropriately classify related work on current approaches in Section III. In the following Section IV, we introduce our concept for tackling the challenges and achieving our objectives with an UAV ensemble. We further illustrate a concrete implementation of our concepts with a prototype in Section V, which we subsequently evaluate in Section VI within several scenarios. Finally, we conclude our findings in Section VII.

* This work is partly funded by the German Research Foundation (DFG) under the COMBO grant.

II. Problem definition

Compared to a single UAV, the increased number of devices involved in the operation of an ensemble of UAVs increases the error susceptibility of the entire system. Therefore a system for monitoring the entire ensemble and reacting to malfunctions is necessary. One way to monitor a group of UAVs is to have one pilot per unit who, if a problem occurs, has the option of manually controlling his UAV to bring it back under control. This procedure scales poorly due to the increasing need for ground personnel in proportion to the number of units. In order for the ground personal to react to malfunctioning UAVs, a way for identifying the correct one in the ensemble needs to be found. Particularly in an experimental environment, faulty software and hardware can lead to undesired behavior, which is why strategies are needed to react promptly and appropriately to defects and malfunctions. Additionally, existing ground control software for UAVs often combines the mission planning and execution software with the emergency control. This approach is not optimal as it requires the safety pilot to execute and monitor the mission as well. A similar separation of safety-critical and operational roles can be found in the architecture of autonomous driving vehicles [6]

In this paper, we present a solution for monitoring an entire ensemble and controlling it in case of a malfunction. Our contribution is a solution to display all data of the ensemble in a clear way without overwhelming the safety operator. If a malfunction is detected, we provide strategies to identify and selecting the malfunctioning UAV. We also present ways to react to malfunctions in appropriate ways. Our approach is merely a safety control and agnostic to the mission control used in order to provide more flexibility to

the task of the ensemble. This also reduces the workload of the safety operator. Task definition and deduction, distinct from this paper, as well as synchronization mechanisms between UAVs in a multi-agent framework were presented in previous work [4].

III. Existing solutions for controlling UAVs

In conventional ground stations for the control of commercially available flight controllers, the status of UAVs can be queried and their parameters can be set in a graphical user interface. Examples are the software *QGroundControl* [7] or *APM Planner* [8]. Autonomous flights can also be planned and monitored using these tools. Often the interface is primarily designed for the use of one or only a few UAVs [9]. Paparazzi [10] is a combination of GCS software, flight controller and associated software. The system, which has been in existence since 2003, was designed for robustness and security from the outset. The correctness of the safety-critical parts of the flight controller code was formally proven and outsourced to a dedicated micro-controller of the flight control system [11].

The Fraunhofer Institute presents the *Karlsruhe Generic Agile Ground Station*, a GCS for various robot and sensor platforms [2], [12]. The system, which is designed for monitoring large or inaccessible areas, is operated by two pilots who are responsible for evaluating the transmitted image and sensor data and for controlling the individual mobile robots. Bürkle et al. [13] present a ground control solution for UAV swarms. Multiple operators monitor safety-critical aspects of the swarm as well as the mission status. The solution of Haque et al. [14] replaces the single laptop running a GCS software with a dedicated micro-controller to handle basic control of the UAV. A connected laptop is only used for the display of telemetry data and weather information. This architecture allows the operator to control the UAV even if the laptop fails.

The plugin for the ground control software *QGroundControl* developed by Dousse et al. [9] extends its functionality by the possibility to control more than 15 UAVs with a clearly arranged user interface. An operator selects the devices to be commanded and controls commands such as landing, an emergency stop or assigning manual control by a pilot's remote. The only task of the pilot is the manual control of the copters assigned to him by the operator. The operator thus takes over tasks for controlling and planning the mission as well as for emergency control of the copters in case of failure. With this approach, when a fault occurs, the operator must tell the pilot which copters he is controlling and the pilot must identify them by means of color markings [9]. For light shows with UAVs, Intel uses the self-developed UAV *Shooting Star*, which is controlled by a central software [3].

IV. Concept

In this section, concepts for controlling and monitoring groups of mobile flying robots are presented. Identifying individual members of a group and displaying their status is often essential for assigning tasks to the UAVs. Therefore, concepts are presented that allow the identification and selection of a single or multiple participants of a group and to visualize the status of the ensemble. In addition, methods will be developed to reduce the time from the detection of an error to the initiation of an action to correct it or even to detect errors before they occur.

A. Overview of the status information of all UAVs

Many problems can be detected by ground personnel even before they occur, if important status information of the UAVs is displayed live and in a concise way. This can be done because the information can be used to draw conclusions about errors that will occur soon. For example, if a pilot detects the low battery status of a UAV at an early stage, he can land it and replace either the battery or the entire UAV with another one before the battery voltage reaches a critical level. This problem can be avoided if all battery states are clearly displayed at runtime. It can also make it easier to identify the device with low battery status. A common solution to this is to attach piezo speakers to the UAVs, which generate an acoustic warning signal when the battery level is low. However this solution scales poorly, because with increasing swarm size the allocation of the warning signal to a UAV becomes more difficult. The display of parameters such as battery status and GPS reception for each UAV in a GCS can, in combination with a possibility to land it automatically, reduce the time until the error is detected and corrected.

B. Identification of individual Robots

In the event of a malfunction in a group of UAVs, the affected participants must be quickly and reliably put into a state in which they do not negatively affect the functionality of the remaining swarm or harm their environment. For this purpose, ways have to be found to identify the malfunctioning UAVs and to make them distinguishable from each other as well as to display their current state (orientation, battery status, etc...) as intuitively as possible for the user.
In conventional ground control stations, the data from a UAV are transmitted by radio to a computer and displayed there in a graphical user interface [11]. In addition to the current position, important data such as GPS reception, battery status and mission data can be viewed on the screen. In the software *QGroundControl* a map giving an overview of the position of individual UAVs is displayed as well as a list of connected UAVs and their status. However, this data is often not presented and arranged clearly which makes it difficult to get an overview of the status of the whole group [9]. Dousse et al. [9] suggest a color assignment in the user interface which depends on a value stored in the UAV. If the physical UAV is provided with a marking of the same color, there is a possibility to connect it with its representation in the user interface. However, this solution does not scale well, because with increasing swarm size, the colors of the UAV become more and more similar and thus a reliable differentiation gets more difficult. A possibility to extend this process is the attachment of LED indicators to the individual UAVs to

provide additional feedback and to allow the pilot to see if the correct vehicles were selected. If different colored LEDs are attached to different places on the UAV, its orientation can be determined. The use of addressable RGB LEDs allows the additional display of different states (for example, blue for *selected*, flashing red for *low battery*).

C. Reduction of the reaction time

If a UAV in the ensemble should get out of control, it is important for the operator to be able to intervene quickly. In order to be able to control a swarm reliably even with an increasing number of participants, control commands need to be sent to several participants at the same time. In many existing ground control stations, control commands are sent to each UAV individually [9]. For each UAV it is necessary to select the UAV itself first and then the respective action. Often a parallel execution of a command is also mandatory. For example, if the UAVs fly in a close formation one after another, it would be fatal if the first UAV stops while the others continue their flight and collide with the first UAV. A synchronous and parallel execution of the commands is therefore essential. Global execution of commands can also be useful in cases like a global emergency stop.

D. Reaction to malfunctions

If the malfunction of a UAV is detected and the correct UAV is identified, it is essential to provide the operator with a means to react to the error. When a UAV is malfunctioning it can be required to stop from moving any further by using *Loiter* (*Position-Hold*) mode. This can be useful if the UAV is flying towards an area where it could damage its environment. If a UAV is in the Loiter mode, it tries to maintain its position in space via the internal position control (see [15]). Disturbing influences, for example by light wind, are compensated by the position controller.

a) Landing: Executing the *Land* action causes the UAV to steadily reduce its altitude in a slow descent without continuing to fly in other directions (see [15]). To prevent drifting and to detect the distance to the ground, it is necessary that the UAV knows its position in space and actively counteracts a position deviation.

b) Return to Home: In *Return to Home* (RTH, also *Return to Launch* / RTL) mode, the UAV first climbs to a height where it cannot collide with obstacles near the ground. It then flies back to a *Home* point previously defined by the pilot, which is usually near the launch site. With this strategy, a UAV which is too far away from the group to be flown back safely manually (e.g. in case of loss of visual contact) can automatically fly back to the ground crew (see [15]).

If the UAV has no possibility to determine its position or if this is disturbed, then its position control no longer functions. The control possibilities are then limited to the following two basic functionalities

c) Full Manual Control: The UAV can be moved in all directions by manual control, i.e. by setting the direction of flight using a remote. However, due to the omission of position control, the UAV can drift and the position and altitude must be maintained manually. This control option can be used as a strategy to safely land a UAV with disturbed position sensors.

d) Emergency Stop: If a UAV becomes completely uncontrollable, an emergency stop can be used to prevent the aircraft from continuing its flight and thus endangering the environment. The function is similar to that of an emergency stop switch in industrial machinery. In the case of the UAV, this means that power to all motors is turned off, which results in an immediate crash. Since the crash is highly likely to cause damage to the machine, this control option is intended as a last resort.

E. Interface for exchangeable mission control software

When developing software to be tested in combination with a group of UAVs (e.g. swarm algorithms, mission flights, connection of the hardware to other systems or further development of existing hardware and software features), errors in software and hardware cannot be ruled out. These can lead to UAVs getting out of control or not reacting to mission control commands. In this case, a safety controller decoupled from the development system can often still intervene and land the UAV safely. In scientific or development scenarios where the mission control software is under heavy development, it is essential to provide a separate system that can intervene on its own communication channel in case the main mission control program fails.

V. Design and implementation

In this section, we present the design and implementation of our ground station *Ensemble Control* (ECO). This GCS allows for a reaction to various malfunctions in the ensemble in a way that minimizes the damage to UAVs and the environment.

A. Ground Control Station

The GCS consists of a tablet PC connected to a radio remote control. UAVs can be selected and deselected using this tablet mounted over the remote control. It is also possible to perform tasks such as landing or an emergency stop. The wireless remote control is used to manually control the selected UAVs after the tablet has transferred control. The design of the GCS implements the reaction strategies defined in subsection IV-D.

The graphical user interface of the GCS was programmed in Java and is shown in Figure 1. On the left side of the screen, a list of all connected UAVs and their status information is displayed. For each UAV, there are buttons that allow the actions *Return to Home*, *Position-Hold*, *Land*, *Manual Control*, and *Emergency Stop* for that UAV to be started from a single user input. The buttons have large, colorful and uniform icons to help the operator find them quickly. The icons also have a white bar at the bottom to give feedback on what action is currently active. A selection of the corresponding UAVs is possible by selecting list entries. Afterwards, a menu on the right-hand side allows the same actions, which are also applicable to individual UAVs, to be

Fig. 1. Graphical user interface of the tablet application: On the left side is a list of all available UAVs with their status information. For each UAV, the color, battery level, status of the position determination and IP and *MavLink* address are displayed.

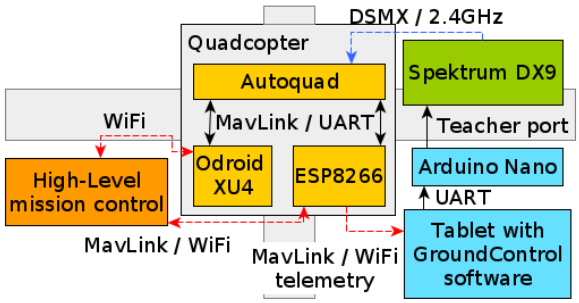

Fig. 2. Extension of the existing hardware architecture.

carried out on all selected UAVs. It is also possible to execute all actions globally, i.e. for every UAV simultaneously. A corresponding menu is also located on the right side of the user interface.

If the pilot wants to influence a single device, for example control it manually, he presses the *Control* button in the list entry of the UAV. If several devices are to perform an action, he first selects them in the list before selecting an action (for example *Country*) in the *Selection Actions* menu. To control all UAVs at once, an action is selected in the *Global Actions* menu. For example, a global emergency stop can be executed by pressing the *Stop* button. In order to avoid accidentally triggering an emergency stop, a pilot must press an emergency stop button twice in quick succession. With the first press a warning signal sounds, with the second press the emergency stop is executed. All other controls also give the user both visual and audible feedback to make the use more efficient.

In addition, the user interface displays the parameters of each UAV that are important for the pilot. Thus the battery level, the status of the position tracking and the Armed / Disarmed status are directly visible for each UAV. Additionally, the parameters change their color depending on their value. For example, the voltage of a full battery is displayed in green. If it drops, the color of the display also changes from yellow to red. This is intended to make it easier to identify critical values. For the transmission of status information the *MAVLink*-Protocol [16] was used.

B. Selection Feedback

In order to quickly assign the individual UAVs to their representation in the graphical user interface, they were provided with colored markers on all sides. The color of the markings is stored in the configuration of the flight controller of the respective device and can be read by the GCS, which then displays this color value in the GUI representation of the UAV (see Figure 1).

In order to give the pilot visual feedback on his actions at the UAV, individually addressable RGB LEDs were installed on each UAV. The LEDs of the UAV fulfill multiple functions. On the one hand, one set of LEDs per arm indicate the orientation of the UAV. Similar to the headlights of a car, the two rear arms shine red and the front ones yellow. Compared to other solutions like differently colored propellers in the front and back, the LEDs provide better visibility, especially at long distances and at night. On the other hand, status information can be displayed directly on the device via additional LEDs. If the status LEDs light up green, this indicates that the UAV is currently being controlled by the pilot or is carrying out an action. When selecting devices, a blue light makes the pilot understand that all blue UAVs are currently selected. If the status LEDs are off, the UAV is neither selected nor controlled. The ESP8266 module filters the LED control packets from the *MAVLink* data stream sent by the GCS and drives the LED modules accordingly.

C. Separate communication channel

The GCS uses a tablet to control an Arduino Nano micro-controller via a serial interface (see fig. 2). The latter generates a PPM (pulse-pause modulation) signal, which is fed into the remote control via the teacher input. The extension makes it possible to overwrite channels of the remote control and thus to forward control signals from the tablet to the UAV.

The control signals of the UAV are coded into channels 7 and 8 of the radio remote control. It only feeds through channels 7 and 8 from the teacher input. For all other channels the stick or switch values of the remote control are used. Channel 7 is responsible for addressing the UAVs while channel 8 represents the action the UAV needs to execute. It is possible to address a single UAV, all currently selected UAVs or every UAV in the ensemble. The action can be selecting or deselecting a UAV or executing one of the reaction strategies defined in subsection IV-D.

To interpret the additional remote control signals the firmware of the flight controller was extended. The UAV checks channel 7 to see if it is currently addressed by the remote. If it is, it executes the action determined by channel 8. The ESP8266 module is used as the return channel for telemetry data. It is connected to the tablet via WiFi, but can still be used for communication with other devices. The separation between emergency control and mission control allows the user to use his own infrastructure for controlling the UAVs independently of the emergency control and to exchange them as required. When designing the GCS, care was taken to not block the use of any means for controlling the

UAV by external programs. The communication interfaces to the ESP module and the Odroid were not used exclusively and are still available for control by external software. This approach is in contrast to most other, which rely on the combination of both functionalities into one unit.

VI. Proof of concept

In the following section the prototype is tested using two scenarios based on real applications for UAV ensembles. First the failure of a single UAV in a building inspection and then the failure of several UAVs carrying a payload are considered. For both scenarios, the reaction time from the occurrence of the fault until the system is transferred to a safe state and the covered distances are measured. For illustration, we provide a video of both setups[1].

A. Inspection of a building with a UAV

A possible application for UAVs is the inspection of buildings [17]. Here, a UAV films the building from different perspectives. In the example considered, a UAV with mounted camera is to automatically patrol parallel to a side of a building at a constant height in order to inspect it. In this scenario there is a no-fly zone, which includes the building and directly adjacent areas. A randomized error in the UAV's autonomous control software causes it to abort its mission at a random time and fly in a different direction causing it to either head for the no-fly zone or move away from it. In order to ensure a safe test environment, the test was carried out on a field without directly adjacent buildings, which was divided into different areas. Along this zone a UAV flies parallel to one side of the building between two points, which are about 15 m apart. It is oriented towards the building to inspect it with a camera. Its speed was limited to 2 m/s for the outdoor experiments. In addition, the firmware of the FC has been extended in such a way that the UAV deviates from this path after an arbitrary time and shows one of the following two malfunctions.

In situation 1, the UAV unexpectedly moves away from the building in a direction where there are no obstacles or people. In this case it is safe to try to bring it back under control. The reaction strategy *Position-Hold* prevents the UAV from flying away. The UAV can then be retrieved using the *Return to Home* function and landed either manually or automatically using the *Land*. In situation 2 the UAV moves towards the no-fly zone. Here the pilot has to decide if the UAV can be brought under control again. If this is not possible, the UAV will continue to enter the no-fly zone during the unsuccessful attempt. In this case, it is generally advisable to prevent the UAV from causing damage to its environment by making an emergency stop directly.

The two situations were tested and both a video and GPS position data of the UAV were recorded. For each situation the worst case scenario was considered, i.e. the drift of the UAV perpendicular to its flight path either into the no-fly zone or away from it. The speed of the UAV was limited to 2 m/s. In the first scenario, the pilot executed the reaction strategy *Stay* after about 1.2 s. The transmission of the signal to the UAV took about 0.8 s. The maximum deviation from the target position was reached about one second later. The UAV then flew back a little and came to a complete stop after another 2.7 s. The total time from the occurrence of the error until the UAV stopped was 5.7 s. In the second case of failure, the UAV flew into the no-fly zone at a maximum speed of 2 m/s after an undefined time and had to be brought to a standstill with an emergency stop. The reaction time of the pilot and the transmission time of the command is very similar to the first situation. Due to the changed reaction strategy, the UAV reached a standstill in less than 3.5 s after the error occurred. However, this was achieved by crashing the UAV. After some lateral movements the UAV breaks out in positive direction of the Y-axis into the no-fly zone. After about 4.5 m it comes to a standstill due to the crash. In comparison to the first situation, the maximum deviation from the target position is increased by about one meter. This behavior can be explained by the fact that in the first case the UAV brakes actively and thus reduces speed, whereas in an emergency stop the motors are switched off and the UAV continues falling in its original direction until it hits the ground.

B. Collaborative transport of a payload with several UAVs

In the second experiment the complexity of the setup was increased by using several UAVs. In this experiment four UAVs cooperatively transported a sensitive cable at a constant height. Four UAVs (Q1-Q4) alternately flew to two waypoints. They carried a cable which can be damaged by excessive tension. For this reason, they synchronized themselves so that they always flew in a line formation while keeping their distance from each other constant. After an indefinite period of time, some of the UAVs stopped due to a synthetically introduced software error, the others continued to fly. In order not to damage the cable, all UAVs first had to be stopped and then manually returned to the line formation to be able to land. The experiment was performed in an indoor flight laboratory. It is equipped with a Vicon camera tracking system and thus allows the exact positioning of the UAVs. The four UAVs were flying at an altitude of about 0.8 m and at a distance of about 1.5 m from each other. The distance between the two waypoints was two meters and the UAVs flew at a maximum speed of 0.45 m/s. The mission control was performed by a test program which was connected to the *Robotics API* [18]. This program let the UAVs fly to the waypoints in turn and made sure that some UAVs stop at a random time. The prototype of the GCS was running in parallel to the mission and allowed the pilot to intervene in case of an error. When the latter happened, the *Stay* action was first performed globally to prevent the UAVs from moving further apart. Then the stopped UAVs could be selected and their position in space could be changed by manual control to restore the line formation. Thus, by reducing the distance between the UAVs, a transported cable could be relieved.

[1] `https://video.isse.de/eco`

Afterwards the action *Land* could be executed globally to land the UAVs synchronously and automatically. Like in the previous experiment, the time and distance from the occurrence of the error to the standstill of all UAVs were measured. The determination of the times was done by video recordings. For the distances covered, the position data of all UAVs were recorded in the *Vicon Tracker* software and then evaluated. The times required for the transmission of the control command average at 0.65 s and are thus very similar to the values of the previous attempts. It is important that the transmission time does not increase with the number of UAVs like it would in existing solutions where the action has to be performed for each UAV individually. The reaction time of the pilot is on average somewhat lower, but varies between 0.6 s and 1.5 s. The reduction of this time span can possibly be explained by the fact that in this attempt the remote control was placed on the table and did not have to be held by the pilot as in the two previous attempts. This left both hands free for the pilot to intervene more quickly. The UAVs covered an average distance of one meter after the error occurred before they came to a standstill. The maximum distance covered by a continuing UAV compared to a stopped one was 1.5 m.

VII. Conclusion

In this paper first problems of solutions for the control of mobile robot ensembles were identified. In addition, existing solutions to the problem were presented. We then presented approaches for monitoring an ensemble as well as for the identification of individual participants of the ensemble. In addition, suitable reaction strategies for error cases and reasonable control possibilities for robot ensembles were discussed. Furthermore we demonstrated our implementation and realization of a GCS for monitoring and controlling several UAVs with only one pilot. This was successfully tested in the form of a proof of concept in two different scenarios with the occurrence of a partially expected error. The benefits that our system provides over preexisting solutions are the clear separation of experimental mission-specific code from safety-critical systems and a fast and user-friendly way to monitor and control a small ensemble of UAVs by only a single pilot.

During the evaluation of the prototype, potential improvements were identified. For example, the workload for the pilot could be further reduced when the swarm is larger by dividing it into sub-swarms, each of which is monitored by one pilot. However, this would require a way to separate the sub-flocks in addition to the individual UAVs, so that each pilot can identify his swarm. Depending on the application, this can be done, for example, by spatial separation of the sub-flocks or by defining different groups that can be quickly selected. Estimating the position of the UAV is especially difficult during outdoor missions. An overview of the overall situation from a bird's eye view, which can be very helpful to show the position of the UAVs. It would also be possible to enter reactions directly into the map and have the program automatically perform an action, for example an emergency stop, when the UAV enters a no-fly-zone. On the other hand, this automation may make wrong decisions due to its limited perception of the environment. Therefore, when automating such a system, care must be taken to ensure that a human operator remains in place to monitor the system.

VIII. Data Availability

The code and results of the proof of concept can be accessed at `https://dl.isse.de/eco`

References

[1] J. Penders, L. Alboul, U. Witkowski, A. Naghsh, J. Saez-Pons, S. Herbrechtsmeier, and M. El-Habbal, "A robot swarm assisting a human fire-fighter," *Advanced Robotics*, vol. 25, no. 1-2, pp. 93–117, 2011.

[2] A. Bürkle, F. Segor, and M. Kollmann, "Towards autonomous micro uav swarms," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 339–353, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10846-010-9492-x

[3] Intel shooting star drone project page. Accessed on: 2020-01-12. [Online]. Available: https://www.intel.co.uk/content/www/uk/en/technology-innovation/aerial-technology-light-show.html

[4] O. Kosak, C. Wanninger, A. Hoffmann, H. Ponsar, and W. Reif, "Multipotent systems: Combining planning, self-organization, and reconfiguration in modular robot ensembles," *Sensors*, vol. 19, no. 1, 2018. [Online]. Available: http://www.mdpi.com/1424-8220/19/1/17

[5] O. Kosak, "Facilitating planning by using self-organization," in *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Sept 2017, pp. 371–374.

[6] A. C. Serban, E. Poll, and J. Visser, "A standard driven software architecture for fully autonomous vehicles," in *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2018, pp. 120–127.

[7] Qgroundcontrol. [Online]. Available: http://qgroundcontrol.com/

[8] Apm planner. [Online]. Available: http://ardupilot.org/planner2/

[9] N. Dousse, G. Heitz, and D. Floreano, "Extension of a ground control interface for swarms of small drones," *Artificial Life and Robotics*, vol. 21, no. 3, pp. 308–316, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10015-016-0302-9

[10] Paparazzi ground control station. [Online]. Available: http://wiki.paparazziuav.org/wiki/GCS

[11] P. Brisset, A. Drouin, M. Gorraz, P. Huard, and J. Tyler, "The Paparazzi Solution," in *MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles*, Sandestin, United States, 2006.

[12] S. Leuchter, T. Partmann, L. Berger, E. Blum, R. Schönbein, and J. Beyerer, "Karlsruhe generic agile ground station," in *Future Security. 2nd Security Research Conference*, 2007, pp. 12–14.

[13] A. Bürkle and S. Leuchter, "Development of micro uav swarms," in *Autonome Mobile Systeme 2009*, R. Dillmann, J. Beyerer, C. Stiller, J. M. Zöllner, and T. Gindele, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 217–224.

[14] S. R. Haque, R. Kormokar, and A. U. Zaman, "Drone ground control station with enhanced safety features," in *2017 2nd International Conference for Convergence in Technology (I2CT)*, April 2017, pp. 1207–1210.

[15] Ardupilot documentation - flight modes. [Online]. Available: https://ardupilot.org/copter/docs/flight-modes.html

[16] Mavlink documentation. [Online]. Available: https://mavlink.io/en/

[17] Industrial skyworks - building inspections. [Online]. Available: http://industrialskyworks.com/drone-inspections-services/

[18] A. Angerer, A. Hoffmann, A. Schierl, M. Vistein, and W. Reif, "Robotics api: Object-oriented software development for industrial robots," *Journ. of Software Engineering for Robotics*, vol. 4, no. 1, pp. 1–22, 2013.