Autonomous Robots (2024) 48:19
https://doi.org/10.1007/s10514-024-10169-1

Bridging the reality gap in drone swarm development through mixed reality

Micha Sende[1] · Christian Raffelsberger[1] · Christian Bettstetter[1,2]

Received: 18 July 2022 / Accepted: 17 July 2024

Abstract
Swarm algorithms promise to solve certain problems in large multi-robot systems. The evaluation of large swarms is however challenging as simulations alone often lack some properties of real systems whereas real-world experiments are costly and complex. We present a mixed reality (MR) system that connects simulated and physical robots though a 5G network, facilitating MR experiments to evaluate communication-based swarm algorithms. The effectiveness of the system is demonstrated through extensive experiments with unmanned aerial vehicles. Measurements show that the communication requirements of swarm coordination are well met by 5G but the computing power of the simulation server can be a bottleneck. However, even when the simulation slows down, communication and coordination take place in real time. In conclusion, 5G-enabled MR experiments are a feasible tool for bridging the reality gap in the development and evaluation of robot swarms.

Keywords Drone swarm · Mixed reality · Reality gap · 5G · Edge computing · Robot operating system (ROS)

1 Introduction

To develop systems with many drones aka unmanned aerial vehicles (UAVs), experimental evaluation for debugging and performance evaluation is crucial. On the one hand, simulations with purely virtual objects can mimic and evaluate many devices in a fast and repeatable manner without posing a safety risk. On the other hand, physical experiments in real environments exhibit higher fidelity and realism. However, physical experiments with multiple UAVs are influenced by a variety of physical, financial, and regulatory framework conditions, over which researchers have little control or which at least represent administrative hurdles. These are, for example, regulatory conditions (flight permission, prohibition of beyond-line-of-sight flights), the availability of pilots, and weather conditions.

Crossing from simulations over the reality gap to physical experiments is a non-trivial task with no general solution (Silva et al., 2016). Some approaches include real-world experiments in the design process (e.g., Koos and Mouret, J- B., Doncieux, S. (2013)), while others focus on creating more realistic simulations to minimize the effect of the reality gap (e.g., Ligot and Birattari (2022)). For bridging the reality gap, we propose MR, which supports real-time experiments of physical and virtual objects mapped into a common environment (Hönig et al., 2015). Step by step, the reality gap can be crossed by gradually increasing the proportion of physical robots as the design process requires and general conditions permit. This avoids the sharp change from simulation-only to physical-only experiments. One can validate a swarm system in realistic experiments and at the same time have a large number of entities.

In this work, we use the term MR in the following manner:

> There are one or more virtual environments containing virtual robots, and a physical environment containing one or more physical robots. All environments share a common coordinate system and are connected via a communication network. The robots can interact by exchanging messages.

We have neither augmented reality (where virtual robots are projected into reality) nor augmented virtuality (where real robots are projected into a simulator).

✉ Micha Sende
sende@lakeside-labs.com

✉ Christian Bettstetter
christian.bettstetter@aau.at

[1] Lakeside Labs GmbH, Klagenfurt, Austria

[2] Institute of Networked and Embedded Systems, University of Klagenfurt, Klagenfurt, Austria

Published online: 30 September 2024

This work presents for the first time an MR system that connects virtual and physical robots via a cellular network to evaluate the behavior of UAV swarms. The system consists of physical UAVs, edge servers simulating virtual UAVs, and communication infrastructure to connect the UAVs. This system enables researchers to evaluate swarms under realistic conditions and increase their size beyond physical limits. It is aimed for the development of communication-based swarm algorithms and demonstration of swarms in the presence of regulatory limitations. The system requires the UAVs to have localization and communication capabilities. As sensor-based interaction is not supported, the UAVs do not need to be projected into a common environment. Nevertheless, the positions of all UAVs are visualized on a terminal, called ground control station (GCS).

Our system focuses on supporting the development of swarm systems where robots interact and apply distributed decision making in order to fulfill a common goal that could not be achieved by a single robot. Such swarm robotic applications are still scarce. Most commercial solutions, such as modern drone light shows, continue to use centralized control and execute pre-defined patterns, despite often using the term "swarm" (Schranz et al., 2020).

To demonstrate its applicability and usefulness in a specific setup, we use the system in a coverage mission, where a swarm of virtual and physical UAVs interact to avoid collisions between swarm members. This demonstration is complemented by performance measurements to evaluate the scalability concerning computational and network resources. We conclude that the used 5G network is well suited for such missions and poses no limitations. However, the performance of the simulation computer limits the swarm size due to high CPU requirements. The swarm performance degrades gracefully when overloading the simulation server: while the movement of the virtual UAVs slows down, communication and coordination still happen in real time.

The paper is structured as follows: Section 2 reviews the state of the art. Section 3 describes the system architecture and Section 4 the protocol and communication architecture. Section 5 evaluates the system. Section 6 concludes.

2 Related work

The related literature can be divided into (1) the application of MR to robotics, (2) the development of communication infrastructures for MR, and (3) approaches to bridge the reality gap.

2.1 Mixed reality in robotics

Applying MR to robotics dates back at least to the 1990s. The main application envisioned was human-robot interaction to simplify the control and supervision of robots (Freund and Rossmann, 1999). This research field is still very active. Challenges include tracking and synchronization of different coordinate systems (Ostanin et al., 2019) and visualization of the robots' intentions (Ostanin et al., 2021). This allows for applications such as debugging of robot behaviors (Hoppenstedt et al., 2019), tele-operation (Welburn et al., 2019; Jang et al., 2021), or shared environments (Phan et al., 2018).

There is little theoretical work on the formalization of MR robotic systems besides a general development process for complex systems that employs multiple levels of virtualization (Jakob et al., 2012). It starts from a purely virtual system and gradually moves to the physical systems by incrementally increasing the amount of physical system components.

System design using MR is an important research direction, as it gives a better understanding of robot behavior (Chen et al., 2009) and facilitates safer testing scenarios (Zofka et al., 2018). Employing MR as intermediate step between simulation and physical experiments gives deeper insight into the system performance and effectively improves the development process of UAV systems (Chen et al., 2012). This leads to a workflow, starting from purely virtual experiments, to MR experiments with some hardware components, to completely physical experiments (Burgbacher et al., 2011). This is closely related to hardware-in-the-loop where simulated robots use some components (e.g., sensors or actuators) of physical robots (Pizetta et al., 2016).

In research on multi-robot systems, MR is proving to be particularly valuable as it enables studies with many robots despite physical, financial, and regulatory constraints. For example, MR can be used to verify the behavior of a UAV swarm, as it was done by Steup et al. (2016). A digital twin of the physical UAV integrates range sensor readings of physical and virtual sensors. The presented approach overcomes the need for an external localization system by using on-board pose estimation relying only on local sensor data. A similar approach is followed by Liu et al. (2020) which present an MR framework for the development of multi-robot systems based on Robot Operating System (ROS). One issue of MR is that virtual and physical robot models are not always identical leading to diverging behavior. A possible solution to this problem is to let the virtual robots match the behavior of the physical ones (Edwards et al., 2018). In later work the authors study stability of swarm configurations influenced by communication delays (Edwards et al., 2020) and colliding flocks (Schwartz et al., 2021). Both studies require a large number of swarm members and are enabled through MR simulations.

2.2 Communication in mixed reality

An important aspect of MR systems is the communication network that connects the robots. A possible architecture

of a distributed MR system was presented by Hamza-Lup et al. (2005) for sensor nodes to share information across a local network. This facilitates spatially distributed collaborative applications. The study presented by Doppler et al. (2017) analyzes the performance of high quality video streams over wireless networks. The authors conclude that the usage of 5G networks will enable the required throughput and latency through massive multiple-input and multiple-output (MIMO) and millimeter wave technologies. To further decrease the communication latency of computationally intensive MR applications, edge computing can improve the performance in presence of high delays to cloud services (Takagi et al., 2019). A comprehensive communication architecture for networking virtual and physical UAVs is presented by Selecký et al. (2018). The authors build their system on an infrastructure-less distributed network where multiple simulated UAVs communicate using a single modem. This creates issues that can alter the validity of experiments. Examples are shared addresses of virtual UAVs which needs to be resolved by specialized routing. Or wrong wireless channel properties of virtual UAVs that need to be modeled explicitly, e.g., the network can partition in a way that virtual and physical UAVs cannot communicate anymore, even though their (simulated) positions are are close to each other.

Our work is related to the solution presented by Selecký et al. (2018) for networking virtual and physical UAVs. However, the major difference is that our solution is built on an infrastructure-based 5G network.

2.3 Bridging the reality gap

The reality gap is a prominent issue in swarm robotics (Dorigo et al., 2021), especially in evolutionary robotics (Silva et al., 2016), where the process of evolving robotic controllers requires many iterations of experimental evaluation. It is probably the main obstacle for progress of evolutionary robotics in real-world applications (Koos and Mouret, J-B., Doncieux, S., 2013) due to the difficulty of accurately simulating physical systems (Matarić and Cliff, 1996). In fact, though Trianni and Dorigo (2006) were able to successfully transfer an evolved controller for communication-based obstacle avoidance to real robots, they were not able to explain the reason why they succeeded.

Several solutions have been proposed for addressing this problem. Embodied evolution evaluates robotic controllers directly on hardware platforms but is limited to small problems since the evaluation can be time consuming (Nolfi et al., 1994; Floreano and Mondada, 1994). This process can be sped up by evaluating only the last few generations of the evolution on hardware or by injecting sampled data from real sensors into simulations (Miglino et al., 1995). Another variant is the combination of simulation and physical experiments to steer the evolution to minimize the performance discrepancy between them (Zagal and del Solar, 2007; Koos and Mouret, J- B., Doncieux, S., 2013).

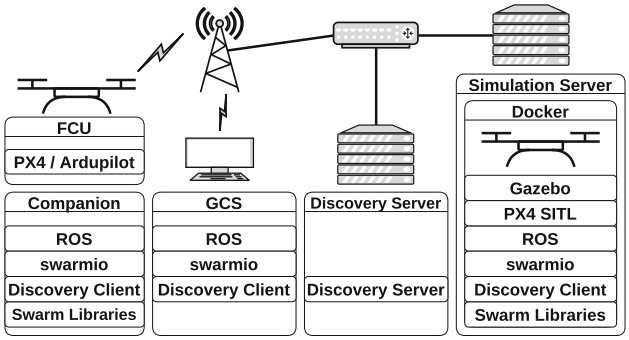


Fig. 1 System architecture

Simulation-based approaches can be applied to more complex problems since they speed up the the evolution through faster evaluation of the generated robotic controllers. Jakobi (1998) introduces the concept of minimal simulation, which is reduced to a base set of features that are relevant for the controller. Since it is non-trivial to derive such a base set, Boeing and Bräunl (2012) instead simultaneously employ multiple simulators in an effort to have the intersection of the simulator features be such a base set. Similarly, Ligot and Birattari (2022) predict the real-world performance of controllers by evaluating them with simulation models, different from the ones used in the design process. Despite these efforts, real-world experiments are still a crucial step in tackling the reality gap.

3 System components

The system consists of three main components visualized in Fig. 1: A swarm of physical UAVs equipped with LTE/5G, one or more edge servers providing both the simulation environment for the swarm of simulated UAVs and the discovery service, and a GCS equipped with wireless communication interfaces. All components are connected via an LTE/5G campus network as part of the 5G Playground Carinthia[1]. The network infrastructure is described in further detail in Section 4.1.

3.1 Physical UAVs

The physical UAVs form the swarm operating in the real world. Each UAV is built from commercial off-the-shelf components. The flight control unit (FCU) is a Pixhawk 4[2] combined with common peripherals, including an u-blox NEO-M8N global navigation satellite system module[3]. The

[1] https://5gplayground.at/

[2] https://docs.px4.io/master/en/flight_controller/pixhawk4.html

[3] https://www.u-blox.com/en/product/neo-m8-series

FCU can run both PX4[4] and ArduPilot[5] firmware. It performs the UAV's low-level control, such as driving the motors to reach waypoints. The companion computer performing the high-level control is a Raspberry Pi 4b[6]. It is responsible for diverse tasks including communication between UAVs, collision avoidance, and mission execution. The UAVs can communicate over LTE/5G. Connectivity is provided by a Quectel RM500Q 5G sub-6 GHz M.2 module connected to the companion computer via USB. The module implements 3GPP Release 15 and supports both 5G non-standalone and standalone mode. It offers 4×4 MIMO in the downlink with up to 2.5 Gbit/s and 2×2 MIMO in the uplink with up to 600Mbit/s.

The companion computer runs Ubuntu Linux. The control algorithms run in the ROS 1 software framework. They are built on top of CPSwarm libraries (Sende et al., 2021). These include swarm algorithms organized hierarchically to separate high-level mission algorithms from low-level, hardware-related ones. The libraries also include the communication library `swarmio` for communication between ROS instances. It contains a bridge that integrates with ROS and serializes proprietary ROS messages into a standardized protocol, which can be deserialized by other `swarmio` instances and forwarded into their local ROS instance. In order for `swarmio` instances to establish communication links, a discovery mechanism is required. To do so, each UAV includes a discovery client that communicates with a central discovery server. Details about the communication follow in Sect. 4.2.

3.2 Edge server

The edge server offers two functions: simulate the swarm of virtual UAVs and provide the discovery mechanism for the UAVs to establish communication between them.

3.2.1 Simulation server

The virtual UAVs are encapsulated in Docker containers. Each container holds the same software components as the physical UAVs (see Sect. 3.1). Moreover, each container includes a simulation environment and an instance of the FCU firmware. Each UAV is isolated from the rest of the swarm in its dedicated simulator, i.e., there is no interprocess communication. The only interaction between UAVs comes from the network communication. This allows for fast deployment of additional UAVs on different server machines and is similar to how the physical UAVs interact. Although the current system operates on a single edge server, the architecture is designed to accommodate the use of more servers, enhancing computational and storage capacities to scale up the number of virtual drones.

The simulation environment used for experiments is Gazebo (Koenig and Howard, 2004). It supports high fidelity mobility in three dimensions and is well integrated with the PX4 FCU firmware for software-in-the-loop execution. In the simulations, PX4 controls a simulated UAV with the inputs provided by the swarm algorithms. PX4 receives the sensor data provided by the Gazebo simulator and forwards them to the swarm algorithms.

3.2.2 Discovery server

The discovery server acts as the central registry that enables distributed communication between the system entities. It maintains a list of available UAVs, informing each other of their existence. Such a central discovery mechanism has certain drawbacks but is required in our LTE/5G campus network as it does not support device-to-device links. In case of other wireless technologies, such as Wi-Fi, a distributed discovery service based on IP multicasting may run on each UAV instead.

3.3 Ground control station

The GCS is the user terminal for the operators of the UAV swarm. It enables them to monitor the swarm by receiving telemetry information from each UAV and to guide the swarm by issuing high-level control commands. As the UAV companion, it runs Ubuntu Linux with ROS, `swarmio`, and the discovery client. `Swarmio` offers command line and application programming interfaces to access the telemetry information diffused by the `swarmio` instances of the UAVs or to transmit control commands to the swarm algorithms. Beyond this, the GCS can communicate with the UAVs using the Micro Air Vehicle Link (MAVLink) protocol, which enables the use of existing GCSs such as QGroundControl[7] or Mission Planner[8] to communicate directly with the FCUs of the UAVs. This can be used to display telemetry data and to override swarm algorithms by manual commands in cases of emergency.

4 Communication

The communication network connects the main system components: physical UAVs, GCS, and one or more edge servers hosting the virtual UAVs.

[4] https://px4.io/

[5] https://ardupilot.org/

[6] https://raspberrypi.com/products/raspberry-pi-4-model-b/

[7] http://qgroundcontrol.com/

[8] https://ardupilot.org/planner/

4.1 Infrastructure

The communication between the system components uses the infrastructure of the 5G Playground Carinthia campus network. It provides a 5G core and a non-standalone 5G radio access network on LTE Band B7 (2600 MHz) and 5G new radio (NR) band n78 (3500 MHz). The small cell radio unit supports 4×4 MIMO with a modulation of up to 256 QAM in the downlink and 64 QAM in the uplink. The integrated antenna provides an output power of up to 10 W and an antenna gain of 10.5 dBi. The campus network is connected to the Internet via a 1Gbit/s symmetrical link but also provides a local breakout to edge servers. All nodes connected to the 5G network can communicate with each other and with the edge servers. While this controlled environment benefits from a guaranteed bandwidth and low latency, it is important to mention that our system architecture also supports commercial mobile networks. The only requirement is that all nodes (the UAVs and the server) receive a public IP address and that the TCP and UDP ports required for node discovery and `swarmio` communication are not blocked.

The virtualized UAVs run in Docker containers, where each container is assigned an IP address. The communication between virtual UAVs is achieved by connecting the Docker containers to the physical network using a medium access control virtual LAN interface that routes data between the Docker containers via the underlying Ethernet interface. Currently, we do not use a channel model to simulate the imperfections of wireless communications. However, a tool such as NetEm[9] could be used to add delay, jitter, packet loss/duplication, or re-ordering to the links between the virtual drones.

4.2 Protocols

ROS 1 was designed as robotics middleware for controlling a single robot. The processes of such a robotic system communicate over a (local) network while always being connected to and managed by a process called ROS master. The master sets up connections between different processes that subsequently communicate directly, using different patterns such as publish-subscribe or request-response. Whereas such a single-ROS-instance architecture already provides networking capabilities, the strong dependency on the master process renders it impractical for multi-robot systems communicating through an unreliable radio network. Our proposed architecture therefore uses a multi-master setup in which each UAV runs its own ROS instance.

`Swarmio` extends the communication capabilities of ROS 1 by bridging the publish-subscribe-based communication of multiple ROS instances. It subscribes to specified

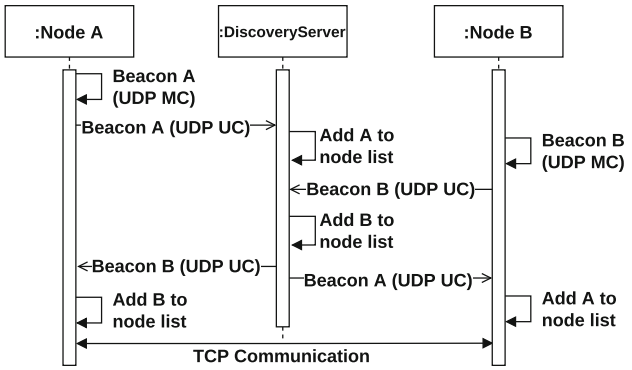


Fig. 2 Node discovery in the network

communication channels, called topics in ROS. Topics are named buses carrying messages of a well-defined data type. `Swarmio` forwards the information received on these topics to other `swarmio` instances, which republish the received information to their local ROS instances. This information can be broadcast to all UAVs or sent via unicast to specific UAVs. The communication between individual `swarmio` instances is based on the Zyre library, which "provides reliable group messaging over local area networks."[10] Node-to-node communication is based on the ZeroMQ Message Transfer Protocol (ZMTP) using TCP sockets. Group management and node discovery is based on ZeroMQ Realtime Exchange Protocol (ZRE) using UDP sockets. A limitation of this protocol stack is that it requires broadcast communication for node discovery. Whereas this is an efficient communication mechanism in some wireless systems (e.g., Wi-Fi), cellular networks usually do not offer this functionality. Thus, we introduce a proxy service on each node and a central discovery server that use unicast communication. The discovery is based on a client-server model as depicted in Fig. 2. The Zyre library broadcasts ZRE discovery beacons. Each proxy forwards the locally received ZRE beacons to the central discovery server. The server maintains a list of all nodes from which it recently received a beacon. It forwards the beacons to each node where the proxy services locally re-broadcast them to perform the discovery with the existing Zyre library.

An independent communication channel is established with MAVLink for the communication between the UAVs' companion computers and FCUs. The FCU sends telemetry to the companion; the companion sends control commands to the FCU. On the companion side, the MAVLink router forwards data packets to ROS, the swarm algorithms, and the GCS. The GCS can thus directly communicate with the FCU to receive telemetry and override the swarm algorithms.

[9] https://wiki.linuxfoundation.org/networking/netem

[10] https://github.com/zeromq/zyre

5 Evaluation

The proposed system consists of multiple subsystems which we evaluate individually before analyzing the whole system. This enables us to account for different aspects with specialized performance metrics. To this end, we first analyze the communication network and the computation requirements individually. We then analyze their interdependence and how both affect the performance of coordination. For this, we employ a mission including virtual and physical UAVs that coordinate through communication. For our performance measurements, we consider a mission in which a swarm of UAVs randomly covers a given area while avoiding collisions. This scenario includes all system components but is simple enough to be the basis for many real-world applications.

5.1 Communication network

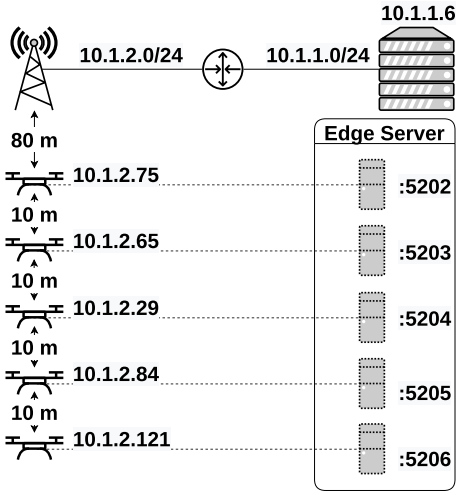
As a first step, we determine the limits of the communication network that consists of both radio link and wired backhaul link connecting the UAVs to the edge server. We evaluate the network performance by measuring the maximum achievable performance in terms of latency (round-trip time) and throughput with the `ping` and `iperf3`[11] tools. As shown in Fig, 3, the experiment consists of five UAVs connected via one 5G base station to five `iperf3` server instances on the edge server. Each UAV runs an `iperf3` client.

Uplink, downlink, and latency are evaluated after each other. The number of UAVs is varied from one up to five. This results in a total number of 15 runs, each run with a duration of 60 s.

Figure 4 shows the average throughput per UAV and the aggregated throughput for up- and downlink. Each data point expresses the mean value. Due to their small size, comparable to the data points, confidence intervals have been omitted for clarity.

Adding more UAVs decreases the per-UAV-throughput (for both up- and downlink), since all share the available 5G NR bandwidth (Fig. 4a). However, increasing the number of communication pairs, first increases the aggregated throughput as a single UAV cannot fully utilize the available bandwidth, before the aggregated throughput decreases again due to the interference caused by the simultaneous communication of all five UAVs (Fig. 4b). The maximum average aggregated throughput is 437Mbit/s in the downlink and 118Mbit/s in the uplink. In our measurements, the round trip time is independent of the number of UAVs with a mean of 9.81 ms and the 95 % confidence interval of 0.27 ms.

As a next step, we evaluate the network requirements of the MR coverage mission with one physical and four virtual



a: Schema of the setup showing spacing of UAVs and communication links (dotted logical links)

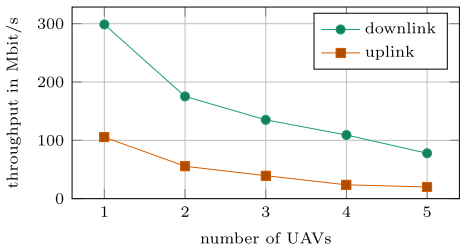
b: Photo of the measurement setup with the 5G base station (gNB) in the background

Fig. 3 Setup for the network measurements

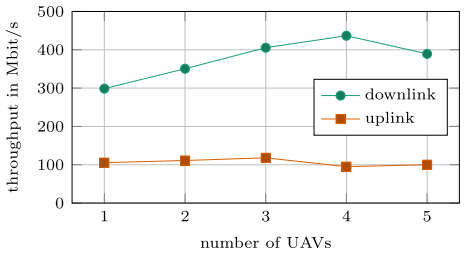
UAVs as depicted in Fig. 5. The physical UAV is connected via a 5G NR link to the virtual drones deployed in Docker containers on the simulation server. Each UAV sends ROS messages, including information about its current position and orientation, to every other UAV. Based on this information, each UAV calculates a route that prevents collisions with other UAVs.

The size of the message payload is 399 B, including overheads introduced by `swarmio`, e.g., for serialization. At an update rate of 10 s^{-1}, each UAV requires about 3.1 kbit/s to inform another UAV about its position. The overall throughput requirement increases with the swarm size N at a rate of $N\,(N-1)\cdot 3.1\,\mathrm{kbit/s}$. Thus, the evaluated uplink bandwidth

[11] https://iperf.fr



a: Average throughput per UAV



b: Aggregated throughput

Fig. 4 Per-UAV throughput and aggregated throughput for downlink and uplink

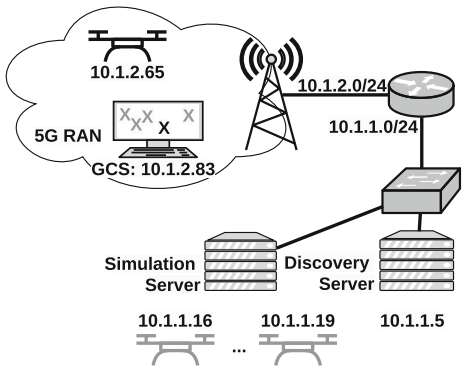


Fig. 5 Network architecture of the MR coverage mission

of 118 Mbit/s can support about $N = 195$ UAVs. This estimation neither includes the overhead introduced by the network protocols for headers, acknowledgments, re-transmissions, etc. nor the uplink bandwidth requirements for any additional payload data.

5.2 Computation

Next, we evaluate the performance requirements of the simulated UAVs. For this purpose, we simulate the missions with a varying number of UAVs and measure the CPU and memory usage. In our system, each UAV resides in its own container with its own simulator. On the one hand, this approach allows for easy scaling of experiments by adding and removing UAVs as well as moving containers between servers in the network. One the other hand, we expect higher resource demands, leading to lower performance compared to the case where all simulated UAVs reside in the same Docker container in a single simulator. Therefore, we compare our system to the latter case with a single simulator. The edge server has an Intel Dual CPU (2× Xeon E5-2698 with 24 cores at 2.3 GHz) with 64 GB ECC DDR4 RAM and 2 TB SSD storage, running an Ubuntu virtual machine with 16 GB of memory and 24 cores.

Table 1 Average memory usage per simulated UAV of the different process classes

	single simulator	multi simulator
simulator	11MB	185 MB
FCU	7MB	7MB
ROS	92MB	190MB
algorithms	303MB	421MB
total	413MB	803MB

We measure the memory usage as the mean of each experiment as it stays mostly constant throughout a whole experiment. There is a linear increase with the number of simulated UAVs for both single and multi simulator scenarios. Table 1 shows the memory used per simulated UAV. As expected, the overall memory usage increases faster for the multi simulator case. The rate is about twice as high for the multi simulator as for the single simulator, where the main difference lies in the simulator processes. Most resources are used by the algorithms, followed by ROS, simulator, and FCU.

As the CPU time increases mostly linearly over time, we provide the relative CPU time. It is calculated as CPU time over experiment duration. Figure 6 shows the CPU usage of the simulation as relative CPU time. Most CPU resources are used by the algorithms, followed by the simulator, FCU, and ROS. As expected, the overall CPU usage increases faster for the multi simulator case. It increases linearly for up to five UAVs before its growth starts to decrease, reaching a plateau, indicating the overloading of the CPU at 4.8 cores per UAV. In the single simulator case, the linear increase of the CPU usage already begins to plateau at around four UAVs or six cores per UAV.

5.3 Coordination

Ultimately, we want to know how well UAVs can coordinate in different situations. Our communication analysis showed that the network provides enough resources for the coverage mission. Our focus question will therefore be: how will overloading the simulation server affect communication and coordination. For this purpose, we measure the real-time factor (RTF) of the simulator(s) during the mission. It expresses how the simulated time slows down in relation to the real wall time. Figure 7 shows the RTF over the number of UAVs

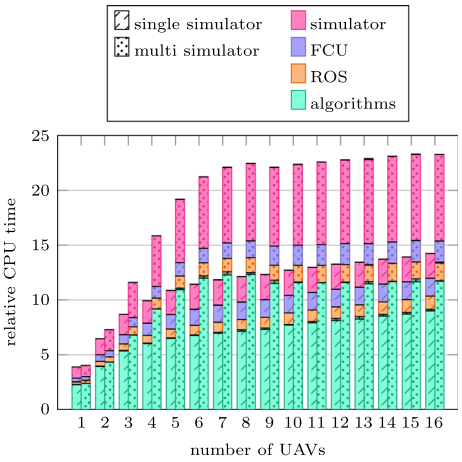


Fig. 6 Relative CPU time used by the different process classes (average and 95 % confidence intervals)

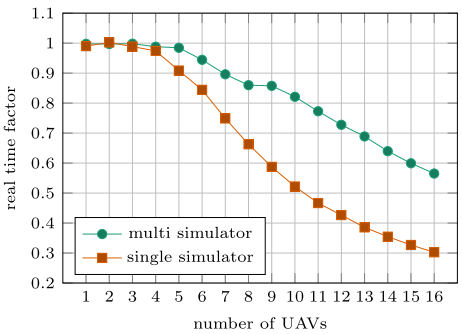


Fig. 7 Average RTF of the simulations

for both single and multi simulator experiments. Due to their small size, comparable to the data points, confidence intervals have been omitted for clarity. We take the mean of each experiment since the RTF stays relatively constant throughout one experiment.

Surprisingly, the RTF is higher in the multi simulator case. From this we deduce that a single simulator cannot parallelize well its computations for large swarms. Hence, the multi simulator is better suited for swarms with more than three UAVs if memory is sufficient on the simulation server. We also see that the performance starts to drop once the growth of the relative CPU time begins to plateau, i.e., for four UAVs in the single simulator case and five UAVs for the multi simulator case.

To understand what these results mean for the coordination between UAVs, we measure the application layer communication delay between UAV pairs. The delay is almost not influenced by the RTF, always staying below 10 ms. Added to the measured communication delay in the 5G network of 9.81 ms ± 0.27 ms, the total communication delay between UAVs always stays below 20 ms. This time is negligible compared to the flight dynamics of UAVs that typically travel at velocities below 20 m/s. Even an extreme deceleration of

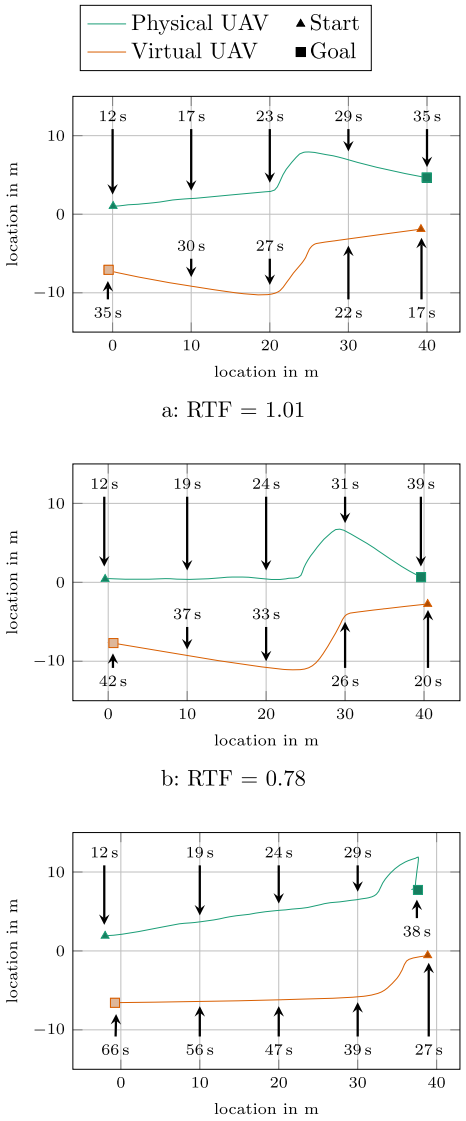


Fig. 8 Collision avoidance trajectories at different RTFs including time after mission start (i.e. takeoff), measured by the physical UAV

10 m/s^2 leads to breaking times that are several orders of magnitude larger than the communication delay.

To better understand the impact of the RTF, we create a benchmark scenario in which two UAVs, one physical and one virtual, directly approach each other. As described above, they exchange their positions and run a collision avoidance algorithm to calculate trajectories that do not collide with each other. We perform this experiment for different RTFs. The results are shown in Fig. 8.

Both UAVs start flying towards each other from 40 m apart. For a RTF around 1 (Fig. 8a), they meet in the middle, avoid collision by changing their path, and continue on towards the goal. When the RTF gets lower (Fig. 8b), the virtual UAV moves slower, since the simulation does not run in

real time anymore. Hence, the UAVs meet closer towards the starting point of the virtual UAV. Nevertheless, they successfully avoid each other. This can be seen even more strongly for a very low RTF of 0.4 (Fig. 8c) where the UAVs meet at the starting position of the virtual UAV. Still they successfully avoid each other and proceed towards their goal. From this, we conclude that a decreasing RTF slows down the movement of virtual UAVs but still allows real-time interaction with physical UAVs.

6 Conclusions

The purpose of our mixed reality system for multi-robot/drone experiments is to facilitate the development of swarm applications by scaling the system size beyond physical, financial, regulatory, human resource, and other limitations. The focus is on UAVs that interact solely by communication: The physical UAVs interact through 5G; virtual UAVs are added in simulation on an edge server running the same ROS-based software stack as the physical UAVs.

An experimental performance study evaluated the feasibility of the network and the edge server simulation. It was shown that 5G is capable of providing the communications infrastructure for a swarm of UAVs to coordinate in real-time. While the uplink of the 5G new radio reaches up to 118 Mbit/s, the required throughput for N UAVs to avoid collisions is $N\ (N-1)\cdot 3.1$ kbit/s. The bottleneck for scalability is the single edge server. The memory and CPU requirements in our exemplary mission scenario are 803 MB and four to five CPU cores per simulated UAV. Nevertheless, if simulations slow down due to limited resources, the system reacts with a graceful degradation: it only slows down the movement of the simulated UAVs but maintains communication and coordination. It is important to note that our architecture is designed for scalability; virtual drones can be simulated across multiple edge servers, effectively distributing the resource load. Thus, the scalability challenge can be addressed by incorporating more edge servers, enhancing overall memory and processing capabilities.

We conclude from these results that the proposed system is well suited to improve the development of communication-based swarms. Through MR experiments it enables researchers to validate swarm algorithms in realistic settings with many UAVs and helps to bridge the reality gap.

A major limitation of the system is that it only works for communication-based interaction. In future work, we like to address this problem through augmented virtuality by projecting the physical UAVs into the simulated worlds. Another direction for future work is the integration of wireless network simulation into the existing framework, specifically for enhancing the realism of communication among simulated UAVs. The goal is to more precisely simulate the imperfections of real-world wireless networks, thus providing more realistic evaluation scenarios.

Acknowledgements This work received funding by the security research program KIRAS of the Austrian Federal Ministry of Finance (BMF) via the Austrian Research Promotion Agency (FFG) under grant agreement no. 879682 (UASwarm). This work was supported in part by the CELTIC-NEXT Project, 6G for Connected Sky (6G-SKY), with funding received from the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility Innovation and Technology (BMK) via FFG under grant agreement no. 891478. All experiments within the 5G Playground Carinthia were funded by the Carinthian Agency for Investment Promotion and Public Shareholding (BABEG). The 5G Playground is operated by BABEG and financed by means of the BMK and the government of the Carinthian provincial.

Author Contributions All authors contributed to setup the projects and acquire the funding. All authors contributed to the basic research idea. Micha Sende and Christian Raffelsberger designed and developed the system and performed the experiments. All authors interpreted the data. Micha Sende prepared the initial manuscript draft and contributed most content. Christian Raffelsberger wrote significant parts of the manuscript, prepared figures, and reviewed and edited the entire manuscript. Christian Bettstetter wrote selected parts of the manuscript and reviewed and edited the entire manuscript.

Funding Open access funding provided by University of Klagenfurt.

Declarations

Conflicts of interest The authors have no competing interests to declare that are relevant to the content of this article. The authors have no relevant financial or non-financial interests to disclose. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.

Research involving human participants and/or animals Not applicable

Informed consent Not applicable

References

Boeing, A., & Bräunl, T. (2012). Leveraging Multiple Simulators for Crossing the Reality Gap. *Proceedings of the International Conference on Control Automation Robotics & Vision (ICARCV)* (pp. 1113–1119). IEEE. https://doi.org/10.1109/icarcv.2012.6485313

Burgbacher, U., Steinicke, F., & Hinrichs, K. (2011). Mixed Reality Simulation Framework for Multimodal Remote Sensing. *Proceedings of the Workshop on Location Awareness for Mixed and Dual Reality (LAMDa).* https://www.dfki.de/LAMDa/2011/program.html

Chen, I.Y- H., MacDonald, B., & Wünsche, B. (2009). Mixed Reality Simulation for Mobile Robots. *Proceedings of the International Conference on Robotics and Automation (ICRA)* (pp. 232–237). IEEE. https://doi.org/10.1109/ROBOT.2009.5152325

Chen, I.Y- H., MacDonald, B., & Wünsche, B. (2012). Evaluating the Effectiveness of Mixed Reality Simulations for Developing UAV Systems. *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)* (pp. 388–399). Springer.https://doi.org/10.1007/978-3-642-34327-8_35

Doppler, K., Torkildson, E., & Bouwen, J. (2017). On Wireless Networks for the Era of Mixed Reality. *Proceedings of the European Conference on Networks and Communications (EuCNC).* IEEE.https://doi.org/10.1109/eucnc.2017.7980745

Dorigo, M., Theraulaz, G., & Trianni, V. (2021). Swarm robotics: Past, present, and future [Point of View]. *Proceedings of the IEEE, 109*(7), 1152–1165. https://doi.org/10.1109/jproc.2021.3072740

Edwards, V., de Zonia, P., Hsieh, M. A., Hindes, J., Triandaf, I., & Schwartz, I. B. (2020). Delay induced swarm pattern bifurcations in mixed reality experiments. *Chaos, 10*(1063/1), 5142849.

Edwards, V., Gaskell, P., & Olson, E. (2018). Calibrating Mixed Reality for Scalable Multi-Robot Experiments. *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (pp. 2183–2185). IFAAMAS. https://ifaamas.org/Proceedings/aamas2018/pdfs/p2183.pdf

Floreano, D., & Mondada, F. (1994). Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural Network Driven Robot. *Proceedings of the international conference on simulation of adaptive behavior (sab): From animals to animats* (pp. 421–430). MIT Press.https://doi.org/10.3929/ETHZ-A-010111549

Freund, E., & Rossmann, J. (1999). Projective virtual reality: Bridging the gap between virtual reality and robotics. *IEEE Transactions on Robotics and Automation, 15*(3), 411–422. https://doi.org/10.1109/70.768175

Hamza-Lup, F. G., Hughes, C., & Rolland, J. P. (2005). Sensors in distributed mixed reality environments. *Journal of Systemics, Cybernetics and Informatics, 3*(2), 96–101.

Hönig, W., Milanes, C., Scaria, L., Phan, T., Bolas, M., & Ayanian, N. (2015). Mixed Reality for Robotics. *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)* (pp. 5382–5387). IEEE/RSJ.https://doi.org/10.1109/IROS.2015.7354138

Hoppenstedt, B., Witte, T., Ruof, J., Kammerer, K., Tichy, M., Reichert, M., & Pryss, R. (2019). Debugging Quadrocopter Trajectories in Mixed Reality. *Proceedings of the International Conference on Augmented Reality, Virtual Reality and Computer Graphics (AVR)* (pp. 43–50). Springer.https://doi.org/10.1007/978-3-030-25999-0_4

Jakob, M., Pěchouček, M., Čáp, M., Novák, P., & Vaněk, O. (2012). Mixed-reality testbeds for incremental development of HART applications. *IEEE Intelligent Systems, 27*(2), 19–25. https://doi.org/10.1109/mis.2012.2

Jakobi, N. (1998). *Minimal Simulations For Evolutionary Robotics* (Unpublished doctoral dissertation). University of Sussex.

Jang, I., Hu, J., Arvin, F., Carrasco, J., & Lennox, B. (2021). Omnipotent Virtual Giant for Remote Human–Swarm Interaction. *Proceedings of the International Conference on Robot Human Interactive Communication (RO-MAN)* (pp. 488–494). IEEE. https://doi.org/10.1109/RO-MAN50785.2021.9515542

Koenig, N., & Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)* (pp. 2149–2154). IEEE/RSJ.https://doi.org/10.1109/IROS.2004.1389727

Koos, S., Mouret, J. B., & Doncieux, S. (2013). The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation, 17*(1), 122–145. https://doi.org/10.1109/tevc.2012.2185849

Ligot, A., & Birattari, M. (2022). On using simulation to predict the performance of robot swarms. *Scientific Data*. https://doi.org/10.1038/s41597-022-01895-1

Liu, Y., Novotny, G., Smirnov, N., Morales-Alvarez, W., & Olaverri-Monreal, C. (2020). Mobile Delivery Robots: Mixed Reality-Based Simulation Relying on ROS and Unity 3D. *Proceedings of the Intelligent Vehicles Symposium (IV)* (pp. 15–20). IEEE. https://doi.org/10.1109/iv47402.2020.9304701

Matarić, M., & Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems, 19*(1), 67–83. https://doi.org/10.1016/s0921-8890(96)00034-6

Miglino, O., Lund, H. H., & Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life, 2*(4), 417–434. https://doi.org/10.1162/artl.1995.2.4.417

Nolfi, S., Floreano, D., Miglino, O., & Mondada, F. (1994). How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics. *Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life)* (pp. 190–197). MIT Press. https://doi.org/10.7551/mitpress/1428.003.0023

Ostanin, M., Yagfarov, R., Devitt, D., Akhmetzyanov, A., & Klimchik, A. (2021). Multi robots interactive control using mixed reality. *International Journal of Production Research, 59*(23), 7126–7138. https://doi.org/10.1080/00207543.2020.1834640

Ostanin, M., Yagfarov, R., & Klimchik, A. (2019). Interactive Robots Control Using Mixed Reality. *Proceedings of the Conference on Manufacturing Modelling, Management and Control (MIM)* (pp. 695–700). IFAC.https://doi.org/10.1016/j.ifacol.2019.11.307

Phan, T., Hönig, W., & Ayanian, N. (2018). Mixed Reality Collaboration Between Human-Agent Teams. *Proceedings of the Conference on Virtual Reality and 3D User Interfaces (VR)* (pp. 659–660). IEEE.https://doi.org/10.1109/VR.2018.8446542

Pizetta, I. H. B., Brandão, A. S., & Sarcinelli-Filho, M. (2016). A hardware-in-the-loop platform for rotary-wing unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems, 84*(1–4), 725–743. https://doi.org/10.1007/s10846-016-0357-9

Schranz, M., Sende, M., Umlauft, M., & Elmenreich, W. (2020). Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*. https://doi.org/10.3389/frobt.2020.00036

Schwartz, I.B., Edwards, V., & Hindes, J. (2021). Interacting Swarm Sensing and Stabilization. *Proceedings of the Symposium on Situation Awareness of Swarms and Autonomous Systems.* NATO STO SCI. https://www.sto.nato.int/publications/STOMeetingProceedings/Forms/MeetingProceedingsDocumentSet/docsethomepage.aspx?ID=44719

Selecký, M., Faigl, J., & Rollo, M. (2018). Communication architecture in mixed-reality simulations of unmanned systems. *Sensors*. https://doi.org/10.3390/s18030853

Sende, M., Schranz, M., Prato, G., Brosse, E., Morando, O., & Umlauft, M. (2021). Engineering swarms of cyber-physical systems with the CPSwarm workbench. *Journal of Intelligent & Robotic Systems*. https://doi.org/10.1007/s10846-021-01430-1

Silva, F., Duarte, M., Correia, L., Oliveira, S. M., & Christensen, A. L. (2016). Open issues in evolutionary robotics. *Evolutionary Computation, 24*(2), 205–236. https://doi.org/10.1162/evco_a_00172

Steup, C., Mostaghim, S., Mäurer, L.,& Velinov, V. (2016). Mixed-Reality Simulation Environment for a Swarm of Autonomous Indoor Quadcopters. *Proceedings of the Rotorcraft Virtual Engineering Conference.* https://www.is.ovgu.de/is_media/Research/rve16.pdf

Takagi, S., Kaneda, J., Arakawa, S., & Murata, M. (2019). An Improvement of Service Qualities by Edge Computing in Network-oriented Mixed Reality Application. *Proceedings of the International Conference on Control, Decision and Information Technologies (CoDIT)* (pp. 773–778). IEEE.

Trianni, V., & Dorigo, M. (2006). Self-organisation and communication in groups of simulated and physical robots. *Biological Cybernetics, 95*(3), 213–231. https://doi.org/10.1007/s00422-006-0080-x

Welburn, E., Wright, T., Marsh, C., Lim, S., Gupta, A., Crowther, B., & Watson, S. (2019). A Mixed Reality Approach to Robotic Inspection of Remote Environments. *Proceedings of the Conference on Embedded Intelligence.* UK-RAS. https://www.research.manchester.ac.uk/portal/en/publications/a-mixed-reality-approach-to-robotic-inspection-of-remote-environments(fd16db8f-3b8b-41b7-953f-8306eb891cbd).html

Zagal, J. C., & del Solar, J. R. (2007). Combining simulation and reality in evolutionary robotics. *Journal of Intelligent and Robotic Systems, 50*(1), 19–39. https://doi.org/10.1007/s10846-007-9149-6

Zofka, M.R., Ulbrich, S., Karl, D., Fleck, T., Kohlhaas, R., Rönnau, A., & Zöllner, J.M. (2018). Traffic Participants in the Loop: A Mixed Reality-Based Interaction Testbed for the Verification and Validation of Autonomous Vehicles. *Proceedings of the International Conference on Intelligent Transportation Systems (ITSC)* (pp. 3583–3590). IEEE. https://doi.org/10.1109/itsc.2018.8569226

Micha Sende received the Dipl.-Ing. degree in computer engineering from the RWTH Aachen University, Germany in 2012 (grade: very good) and the Dr. techn. degree in information technology from the University of Klagenfurt, Austria in 2021 (passed with distinction). He studied wireless communication networks and distributed coordination of mobile robots. He joined Lakeside Labs GmbH in Klagenfurt, Austria in 2017 as researcher in international projects on swarm robotics and cyber-physical systems. His research contributions are in real-world deployment of multi-robot systems with a focus on communication and coordination.

Christian Raffelsberger received the M.Sc. and the Ph.D. degrees in computer science from University of Klagenfurt, Austria, in 2010 and 2015, respectively. From 2010 to 2015 he was part of the Multimedia Communication research group at University of Klagenfurt, where he worked as project assistant in several international research projects. Since 2015 he has been a senior researcher and project leader with Lakeside Labs GmbH, Klagenfurt. His research interests include wireless networking, especially for disaster response and multi-drone systems.

Christian Bettstetter is professor and head of the Institute of Networked and Embedded Systems at the University of Klagenfurt, Austria, and founding scientific director of Lakeside Labs, a research and innovation company. He holds a Dr.-Ing. degree (summa cum laude) in electrical engineering and information technology from Technische Universität München, Germany. His research contributions are in wireless and self-organizing systems with applications in telecommunications and robotics.