Human Factors in Robots, Drones and Unmanned Systems, Vol. 93, 2023, 9–18
https://doi.org/10.54941/ahfe1003742

Third Person Drone – Gamification Motivated New Approach for Controlling UAV

Dennis Lindner, Thomas Hofmann, and Philipp Lensing

Hochschule Osnabrück - University of Applied Sciences, Osnabrück, NI 49090, Germany

ABSTRACT

Over the last years drones became a more and more popular solution for inspection and survey tasks. Controlling these drones, especially in tight spaces, using 'line of sight' or a 'first person' view from the perspective of the drone can be a difficult task. Often users experience an increased rate of difficulty that can be traced back to the limited situational overview of the user. To investigate whether a different form of visualization and interaction might result in a higher level of usability for the user, an experimental workspace was set up, with the goal of exploring the possibility of implementing a 'third person view' metaphor, like one used in video games. To further allow the user to experience his environment the use of virtual reality was used to stream the follower's perspective directly to the user's headset. This allowed the user to fly inside a simulated environment allowing for a control- and repeatable testing ground of the software. The workspace consisted of a simulation in which a 'proof of concept' was developed. In this simulation a drone used a conventional GPS sensor to follow a human controlled drone, offering his view, from a static camera, as a third person perspective to the controller using a virtual reality headset. Within the framework of the project, two aspects in particular were investigated: The performance of the technical system and the basic user experience and ergonomics of this form of interaction. To evaluate the performance of the follower system, the GPS position, as well as execution times and latencies were recorded. The user experience was evaluated based on personal interviews. The results show that the developed system can in fact follow a drone based on the GPS position alone, as well as calculate the desired positions in a timely manner. Yet, the existing delay in movement induced by the controller execution, as well as the drone's own inertia did not allow for continues camera tracking of the drone using a static camera. This introduced several issues regarding tracking and impacted the user experience, but still showed that such a metaphor could in theory be implemented and further refined. The personal interviews showed that users would try tracking the drone by moving their head, like they are used to in virtual reality games. Ultimately, it was deduced that introducing a vector-based drone movement, additional range detection sensor, as well as a moveable camera, controlled via head movement would be next steps to improve of the overall system. Since the prototype created in this paper only contained a bare bones user interface and experience the use of a usability study has been foregone in exchange for a more stable software solution. This allows further research into this topic the possibility of evaluating possible types of spatial user interfaces, which could improve the user immersion.

Keywords: Human factors in human-robot-interaction, Drones and unmanned systems, Virtual reality, Simulated environment, Proof of concept, UX

INTRODUCTION

In the last few years, drones have developed similarly to other older technologies such as the personal home computer. Production, availability and the number of producers increased leading to more affordable drones. At the time of writing some drone models can be bought for less than €200. This affordability and availability lead to drones finding their way into a diverse range of applications. These range from specialized drones for drone racing and hobby drones to industrial task like inspections of buildings like bridges or data gathering for mapmakers (Bandrova & Konečný, 2016). Controlling these drones is achieved by at least one of three ways. The first one being the autopilot, in which the drone itself controls its movement towards a target goal. Secondly a user can manually control the drone via controller using the visual line of sight metaphor (VLOS). In this case the user requires a direct line of sight to the drone to manually control it. Lastly the user can use a camera mounted on the drone, supplying the user with a so called First-Person-View (FPV) or Egocentric-View (EV), to control the drone. In most cases the Autopilot offers a stable and controlled flight but often lacks precision, requires ahead planning of flight paths and doesn't allow the user to execute on the fly changes. VLOS on the other hand offers a Third-Person-View (TPV) of the drone allowing a 360° awareness of the drones surroundings. This effect starts diminishing, the further the drone is located from the operator. By using the FPV metaphor the operator can see from the drone's perspective, as if flying themselves. This allows drone operation in areas where the user can not directly see the drone. The downside of this metaphor is the limited view the operator experiences through the camera reducing spatial awareness and a higher risk of hitting objects in the near vicinity. Given the existing manual control options the question emerges if the two existing metaphors advantages could be merged into a new metaphor. Video games sometimes offer such a perspective to the player by placing the camera behind the user. In this perspective, the user sees the world from a camera positioned behind his character. As such, the user can perceive everything around his character immediately, without the need to turn and scan his environment. This paper tries to answer if such a view can be implemented by using an autonomous second drone following the controllable drone and sharing its view with the user as seen in Figure 1. This view is then shared to a commercial virtual reality (VR) headset.

To achieve this a system of interconnected modules working in tandem to control a drone in a manner that allows it to follow another drone was created using the simulation environment gazebo (Gazebo, 2022), PX4-Autopilot (PX4 Open Source Community, 2022) and the Robot Operating System ROS (Open Robotics, 2022). Due to the use of drone sensor data already created by the environment it offered the possibility of evaluation by collecting and refining this data. To rate the behavior of the drones, the GPS position, as well as execution times of the main cycles and latency between message transfers were measured. This data was then recorded at a centralized location for later evaluation.

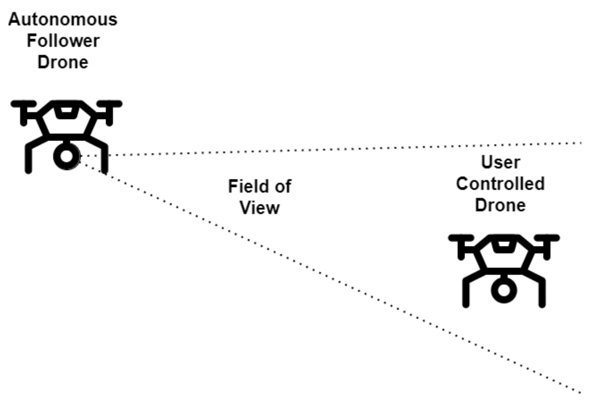

Figure 1: Autonomous drone supplying third person view of controlled drone.

To evaluate the VR application, short personal interviews were carried out with a small number of testers. A Tester was given a running VR application, in which he/she can control the leader drone from the perspective of the follower drone. The subject was then able to provide feedback on handling and possible feelings of nausea. Due to the small part the VR application played in this paper, only a small number of tests was carried out and instead of fixed questionnaires a on hands interview was used to collect results.

CREATING A PROOF OF CONCEPT THIRD PERSON DRONE

An architecture was developed outlining the functionality and required modules as seen in Figure 2. Two drones were created using PX4 and instantiated into a running Gazebo environment. Inside the PX4 environment several

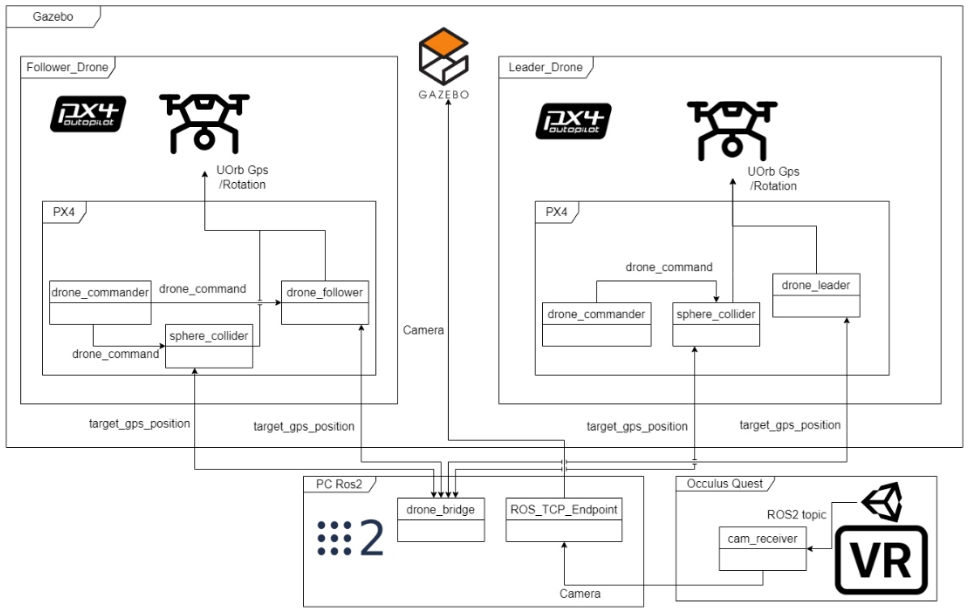

Figure 2: Proof of concept architecture.

modules were created, mainly the drone follower, drone commander, drone leader and sphere collider. Each module serves to realize a specific task. The drone leader polls his position and orientation information in regular intervals and sending them over a network to the follower drone. The drone follower module uses this information to calculate the follower drones desired position and gives the drone commander a move order to its new destination. The sphere collider module observes both drone positions in real time and creates a safety sphere around the drone. In case of one drone breaching this safety area the drone commander will seize to accept move order until the airspace is safe again. All modules interact with each other inside the PX4 environment using a micro-object-request broker, as well as MicroRTPS (PX4 Open Source Community, 2022) to communicate between modules on a different drone. This is done using an offboard ROS2 node aptly named drone bridge, which allows for the exchange of GPS and orientation information. Lastly the camera view was implemented using the inbuild Gazebo camera sensor. The images are read by a TCP server and sent to a client inside a unity environment to be displayed on an Oculus Quest. The images are transferred uncompressed as a raw format. Using equation 1 the required bandwidth B without compression for the image transfer can be calculated, whereby w represents the width of the image, h the height, p the number of bits per pixel and r the number of images send each second. Using an image size of 640x480, a 24bit color

equation 1: Bandwidth calculation for raw image

$$B = w * h * p * r \tag{1}$$

encoding and a framerate of 30 frames per second the potential bandwidth of this image transfer is 2764800 Byte/s, which is roughly 22.1184 Mb/s. An image compression algorithm like JPEG (Wallace, 1992) or the use of a video coding standard like H.264 (Dominguez, 2014) would lower the required bandwidth, but were not realised in this proof of concept.

INTERNAL AND DRONE COMMUNICATION

The communication between internal modules is realized using a micro-object request broker (uORB) introduced by PX4. This allows the asynchronous exchange of messages between modules. To communicate between drones these uORB's are converted to MicroRTPS messages and send over the local network. This exchange can be seen in Figure 3. **Figure 2** showing the internal module communication and the external drone communication using a relay and ROS2.

To communicate the GPS position of the leader drone to the follower the external communication system is used. The message contains a timestamp of when the message was built as well as a timestamp for when the GPS position was last polled. Furthermore, it contains the latitude, longitude, altitude as well as the orientation and current velocity of the drone. The Position is encoded in degrees, while orientation uses Euler angles and velocity uses the North-East-Down (NED) earth-fixed vector to transmit the drone's velocity in meters per second.

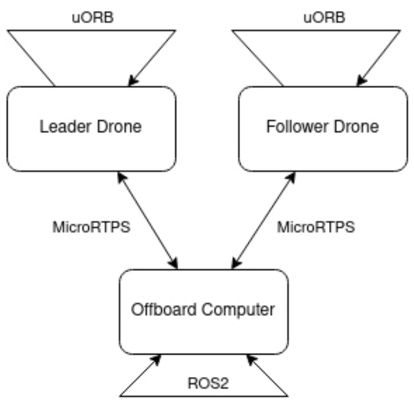

Figure 3: Internal and external drone communication.

CALCULATING THE FOLLOWER POSITION

The movement calculations are executed every cycle for the follower drone. This allows for frequent position updates. Before the calculations are done several tests are performed that ensure the availability of the follower's internal, as well as the leader's GPS data. The new position was then calculated using four steps:

- Calculating the true yaw-rotation of the leader drone.
- Calculating a point on a circle in cartesian coordinates.
- Adding a heading offset.
- Converting the point to GPS-Coordinate (Berger, 2021).

THIRD PERSON VIEW IN VIDEO GAMES

The use of third-person perspective in video games has a long history, dating back to the early days of gaming as seen in Figure 4, from the first arcade

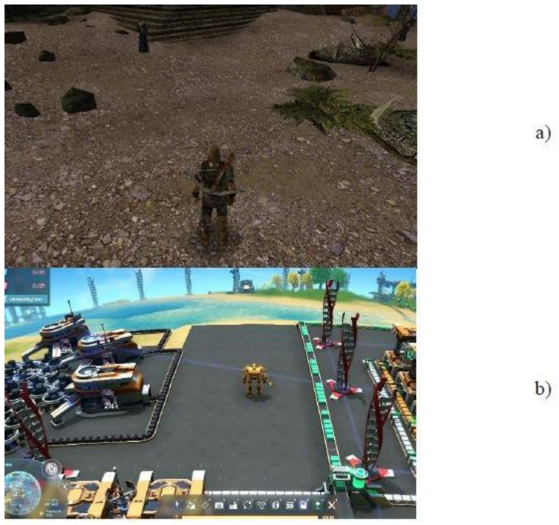

Figure 4: Third-person perspective in video games a) Gothic 2 (2002) & b) Dyson Sphere Program (2021).

games in the 1970s to the large, open-world games of today. Third-person perspective games have undergone many changes throughout the years, with the advent of 3D graphics, motion capture, advanced physics, and AI helping to create more realistic and engaging game worlds.

Advantages that we aimed to achieve from the use of third-person perspective were a greater sense of immersion due to transmitting a sense of movement and interaction from the drone's perspective to the user. Additionally, we wanted to provide a better spatial awareness making it easier for users to navigate and understand the surroundings of the drone.

COLLECTED DATA

To evaluate the practicability of the designed system a flight plan was created for the leader drone to follow. The flight plan consists of 17 waypoints. One Launch point, one land point and 15 move points covering several left, right turns and altitude changes ranging from 10m up to a height of 20m. The data collection process begins once the leader and follower systems have been triggered and the follower positioned itself behind the leader. This flight was repeated two times using a maximum flight speed of 2 and 4 m/s. To compare both drones' behavior and flightpath it was decided to collect GPS data, execution times of the follower and leader modules as well as the latency between sending and receiving of messages. The collected GPS data shows the follower drone reacting to changes in directions as seen in Figure 5 and the follower trying to adjust its position towards the leader.

The reaction times estimated using the graph show a varying degrees of reaction time ranging from $850ms$ up to $2810ms$. Further observation shows the follower reacting delayed to sudden direction changes of the leader. This can be closely observed by calculating the differential distance between the follower and leader across the flight plan. Differences in distance between two drones always follow a change of direction. In detail, once the leader drone starts its first altitude climb a dip in the altitude distance between both drones can be observed. This results in the follower drone closing his altitude difference from the desired 2m to roughly 1m which in some cases could lead to potential collisions. This behavior can be observed for every climb and exists in an inverted state for each drop, where the follower drone ended up above its target height. In between these height changes the follower drone then receives enough time to correct its height position. A similar behavior can be seen when observing latitude and longitude. The observed execution times have shown that the follower and leader work at a stable execution time. The follower module on average takes 9.38μs to complete one position update. The recorded latency has shown that a message from the follower to the leader travels on average 3740.54μs. This is due to the fact these messages do not get transferred on the onboard system but instead are sent over the local network using microRTPS and ROS2. The average total processing time of one position update for the entire system was calculated to be 7520.04μs or $7.52ms$.

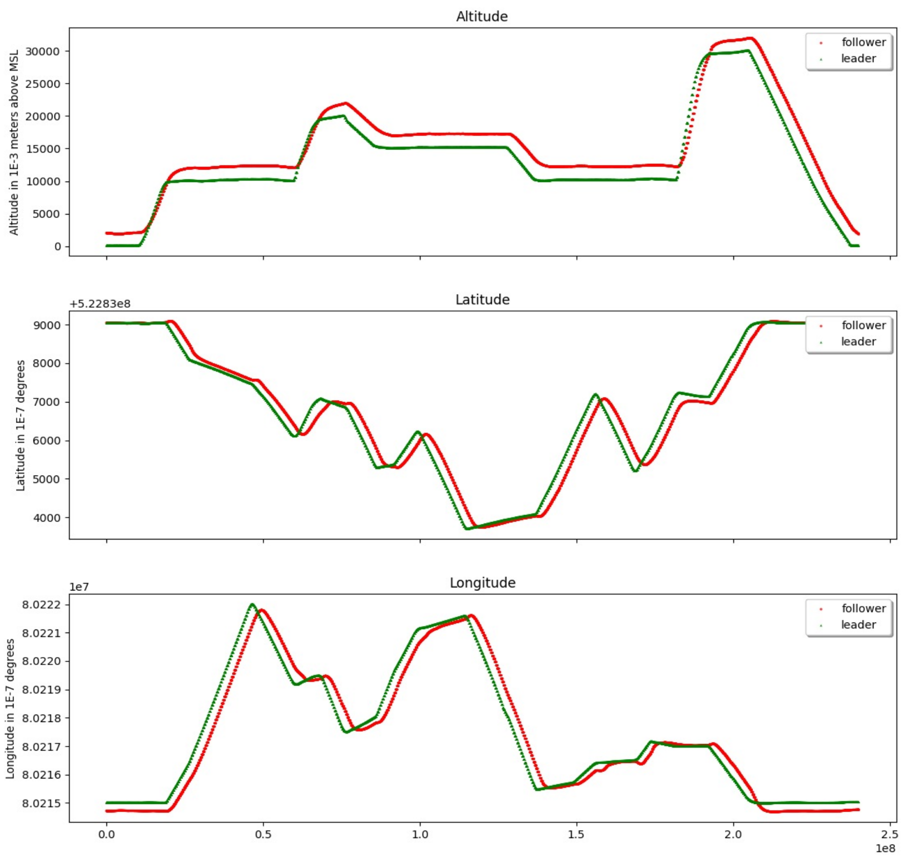

Figure 5: Collected GPS data on test flight using a speed of 2m/s an altitude difference of 2m and a desired horizontal distance of 2m.

USER EXPERIENCE

To evaluate the VR application testers were given the oculus quest running the application and an Xbox controller, with which to control the leader drone. Both drones in this test were set up before giving control to the user and had already taken off. The test person was then allowed to control the drone any way they liked. The fixed camera position gave the users a lack of freedom. Users expected to turn the camera when moving their heads. Additionally, users were trying to track the leader drone by moving their heads. The video quality of 640x480 pixels used in the tests turned out to be acceptable but should be improved to increase the number of details visible. After a short introduction to the control scheme used on the Xbox controller users were quickly able to fly the drone even without prior experience and the need to see the controller, which wasn't possible due to the user being inside a VR environment and the controller not being tracked. In total the overall acceptance of the environment for the user was low. This was mainly due to the low quality of the environment being offset by the expectations of the testers, which were used to higher quality environments offered by most video games today. This in turn begs the question if in today's age solutions regarding gamification in real world applications need to be of high quality to be even accepted by the broad majority as a suitable solution. For future endeavors into gamification this could mean that several ideas already tested or in development might simply fail due the low quality of the developed environment.

CONCLUSION

The observed data indicates that the follower module is in fact able to follow the leader. Additionally, the total message propagation time of the system measured allows for quick course corrections. Nevertheless, the drone required a high amount of time to execute the new waypoint. collecting the GPS sensor data made it possible to directly see the performance of the follower and showed clearly the difference in both drone's pathing. The collected execution times and latency offered an acceptable view into time dilation inside the system but suffered from several outliers and inconsistencies not fully explainable with the collected data. Additionally, the GPS sensor sufferers from several internal noise factors. These factors are present in the simulation as well and might contribute to inaccurate data. When transferring the system to real drones deciding on the type of GPS used, the location, as well as time of year, will end up directly affecting the quality of the

recorded data. As such we recommend the use of real-time kinematic positioning systems.

Additionally, only a small part of this paper focused on collision avoidance. In a real application, it is important to avoid collisions of the follower drone not only with the leader drone but also with objects like buildings, trees, or even humans. A more sophisticated collision avoidance system would be necessary to reduce the risk of these types of collisions. While potential collision of the follower drone should be handled automatically, collisions regarding the leader should be visualized for the user to act accordingly.

Evaluating the results of this paper we propose several improvements for future work regarding the implementation of such a system. The use of a smaller follower drone could be explored. These so-called mini drones can fit in the palm of a hand and usually weigh around 20g. Their low weight could make them ideal to reduce the movement delay. The downsides of these mini drones are their vulnerability to wind, as well as their low flight times. Regarding the used network bandwidth transferring the camera images to the oculus quest we propose the use video encoding and compression standards like H.264.

To reduce camera-shake and allow the user to control the camera using head movement a camera that can be controlled should be used. This could be used to not only move the camera in the user's looking direction but also stabilize the camera image resulting in a less shaky video feed. The use of a physical gimbal attached between the drone and camera is also recommended.

Enhancements to the User Experience (UX) could be made. This includes the addition of sound to the system. This decision is based upon the human expectation of hearing sounds matching the surroundings. Sound could be used here to further enhance the immersion of the VR application. A rotatable camera incorporating the users head movement might further improve the immersion of the system, by allowing the user to change his view by simply turning his head. Furthermore, the UI could be enhanced by adding information to the interface. These could include a heading compass, which shows the heading the drone is moving towards in degree, and the current speed, as well as an artificial horizon displaying the yaw and roll of the drone. Important here is to display time-critical information in a way that does not distract the controller. A good starting point for such a UI can be found in the gaming scene, where spatial UIs are already established. Spatial UIs offer a great addition regarding increased immersion, by making the user feel like he is interacting with a real 3D environment rather than a flat screen. The video gaming industry already uses these types of UI. As an example a possible transfer from gaming to real world applications could be a so called non-diegetic spatial UI. Non-diegetic spatial UI refers to a type of user interface that is not part of the game world, or "diegesis," and is instead presented to the player as an overlay or separate display. A potential use for such a UI could be the highlighting of objects for the user. These objects could also be located behind walls or buildings offering the user a sort of X-Ray vision.

Testing drones using higher speeds, than those used here, should be considered.

In addition to recording GPS data, it should be considered to record more relationship data between drones. This could encompass a real-time distance measurement. This can be done either based on an image recognition system or using distance measurement sensors like light detection and ranging (LIDAR).

To improve on the collision system a different philosophy, in which the follower actively tries to avoid collisions should be considered. This system could be based on several possible sensor candidates. Possible candidates include ultrasound, millimeter wavelength, laser, or LED sensor, of which all have different advantages and disadvantages.

Given jobs like surveying or inspecting are part of what drones can be used for, it is clear to see that, especially in small spaces, a view that allows the controller to see his drone and fine control his movement from a third-person perspective can be an advantage. A third person view could become an easier way to move drones through a forest, evaluate tree growth, or even be used in applications like inspecting large pipelines from the inside, without the user requiring expert skills in drone handling. Furthermore, the requirements for the follower drone are small enough so, that smaller, more agile drones could be used as a follower. This in theory could improve the delay issues the follower experienced in this thesis. The minimal required features, such a drone would need, include a camera and the ability to communicate with the leader drone. The technology of a follower perspective can also be expanded on by using it in other applications, where the follower does not follow a drone but a different object to offer a new view. Trucks could use this system to eliminate blind spots or improve parking functionality.

Future work on this topic should focus on three aspects. Firstly, different types of follower movements should be considered, tested, and evaluated. The next aspect to focus is the already mentioned collision avoidance. Lastly a usability study regarding possible UI concepts for a VR-based drone control application should be devised.

Answering the initial question, whether an exocentric view can be implemented on a drone system using virtual reality we conclude that a third-person view can be implemented on drones with today's camera, GPS, and commercial VR technologies. Furthermore, we found a wide range of possible applications a third-person view could be used. Additionally the paper revealed that such an application requires to be of a relatively high degree of quality, to be fully accepted by the user as a potential alternative to the existing metaphors. This shows how in today's day and age interfaces and applications need to constantly evolve further to meet the demand of the users. This demand can easily lead to what in the software industry is referred to as feature creep resulting in bloated and over-complicated designs. This demand could stand in the way of creating a usable and safe UI design for a third person control system. Further research should as such consider the high requirements of the users from the beginning while keeping a clean spatial design with distinct and basic functionality that does not distract the user from his ability to safely control the drone.

REFERENCES

Bandrova, T. & Konečný, M., 2016. *6th International Conference on Cartography & GIS*. Sofia, Bulgarian Cartographic Association.

Berger, C., 2021. *WGS84toCartesian - a simple header-only, single-file library to convert WGS84 positions (latitude/longitude) to/from Cartesian positions using Mercator projection for C++..* [Online] Available at: https://github.com/chrberger/WGS84toCartesian [Zugriff am 07 09 2022].

Dominguez, H. d. J. O. a. V. O. O. V. a. S. V. G. C. a. C. E. D. G. a. R. K., 2014. The H.264 Video Coding Standard. *IEEE Potentials,* 33(2), pp. 32–38.

Gazebo, 2022. *Gazebo - Robot simulation made easy.* [Online] Available at: https://classic.gazebosim.org/ [Zugriff am 30 08 2022].

Open Robotics, 2022. *ROS - Robot Operating System.* [Online] Available at: https://www.ros.org/ [Zugriff am 03 01 2023].

PX4 Open Source Community, 2022. [Online] Available at: https://px4.io/software/software-overview/ [Zugriff am 31 08 2022].

PX4 Open Source Community, 2022. *PX4 - RTPS/DDS Interface: PX4-Fast RTPS (DDS) Bridge.* [Online] Available at: https://docs.px4.io/main/en/middleware/uorb.html [Zugriff am 31 08 2022].

Wallace, G., 1992. JPEG. *IEEE Transactions on Consumer Electronics,* 38(1), pp. xviii–xxxiv.