2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)
Bari, Italy. October 6-9, 2019

High Precision, Intuitive Teleoperation of Multiple Micro Aerial Vehicles Using Virtual Reality

Robert Ladig, Kazuhiro Shimonomura

Abstract— **In this work, we present a method for the individual, simultaneous teleoperation of multiple multirotors. Instead of using a traditional four axis remote controller to steer the small sized quadrotors used in this work, all degrees of freedom of the device are intuitively controlled by a human operator in virtual reality, allowing high precision maneuvering of the vehicles through obstacles in three dimensional space. The proposed method enables the human operator to precisely control multiple micro aerial vehicles simultaneously, without the use of formation flight or other grouping methods. The teleoperation control performance of this approach is compared to a traditional four axis control method in several experiments.**

I. INTRODUCTION

A. Overview

Small size aerial vehicles have become increasingly popular in recent years. Not only are they popular as a toy or a simple means to take aerial photographs and videos, but the use of micro aerial vehicles (MAVs) has also picked up in industrial fields as a means to conduct inspections, surveillance or even to collect harvesting data for agriculture [1].

Most commercial, industrial or academic aerial robotic systems available today require the input and supervision of an experienced pilot through a 4 axis remote controller. While the current trend is moving towards fully autonomous aerial platforms, there is always the possibility of false calculations or navigational mistakes that could endanger the system or its surroundings. It is therefore important to be able to initiate teleoperation control that is able to overtake complete control of the system, since the alternative, a system halt, means crashing to the ground and possibly damaging the multirotor platform.

Other use scenarios always require manual control by a human operator, such as collecting training data for deep learning algorithms, evaluating flight routes or if the deployment scenario simply requires manual control due to its complexity.

The most common method of teleoperation for remote-controlled aerial vehicles today, the 4 axis remote controller, is hard to learn and requires a trained operator to execute precise flight maneuvers. It also hasn't been improved upon since 1926, long before the widespread use of multirotors [2]. This makes the need for an alternative teleoperation method paramount and was the guiding motivation for the teleoperation method presented in this work (Fig. 1).

B. Traditional multirotor control

Traditionally, the six degrees of freedom (DOF) that are present in an aerial vehicle (position and heading in 3D space) are controlled by two analog joysticks. When controlling a fixed wing system, 4 DOF (pitch, roll, yaw, thrust) can be directly controlled by a human operator. One joystick controls elevator (pitch) and ailerons (roll), analogous to the control stick that has been used to control planes since the early 20th century [3]. The other stick simulates the use of a thrust lever (air speed) and rudder paddles for the flaps (yaw).

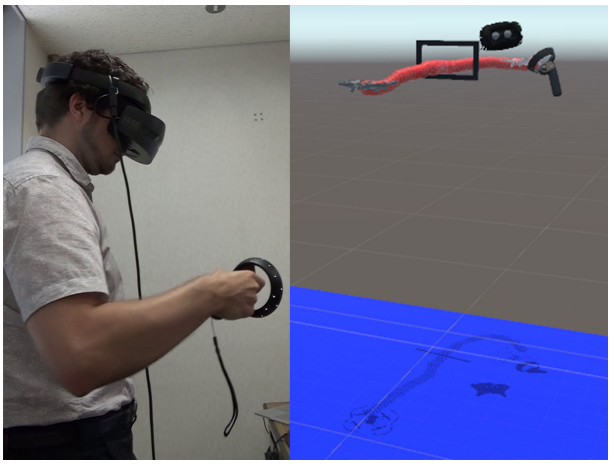

Fig. 1. Proposed control method in real life (left) and in virtual reality (right)

The same interface consisting of two joysticks is used for the kinematically widely different system of a multirotor, which is the most commonly used type of micro aerial vehicle in industry and research, due to its VTOL capability, low cost and maneuverability. There are several control modes that can be used to control a multirotor. In what is most commonly called "manual mode", the operator directly controls the roll, pitch and yaw rate of the device, as well as overall rotor speed. Although this enables high maneuverability, due to the extremely unstable flight characteristics of a multirotor, this mode is mainly used in scenarios where speed is more important than flight stability and safety, like in drone racing. In the most commonly used "attitude hold" mode, the attitude is stabilized by the on-board inertial measurement unit (IMU) and flight controller in a neutral hover position. The human operator then controls 4DOF (position in space, yaw heading) directly. There are many different stick mappings for this control mode, depending on personal preference, but in the mapping chosen in this work, one joystick is used to control horizontal X and Y position of the multirotor, while the other controls Z or flight height of the device and yaw heading (see Fig. 2).

[1]Department of Robotics, Ritsumeikan University, 5258577 Kusatsu, Shiga, Japan ladig@fc.ritsumei.ac.jp

978-1-7281-4569-3/19/$31.00 ©2019 IEEE

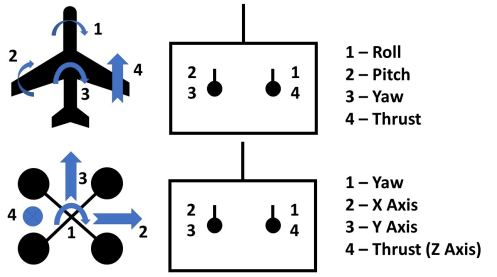

Fig. 2. Traditional control scheme for teleoperated planes and multirotors (in attitude hold mode)

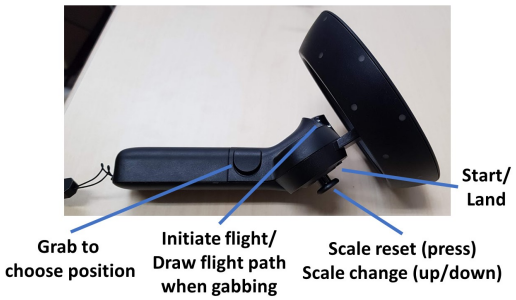

Fig. 3. Button mapping of the VR hand controller used in this approach.

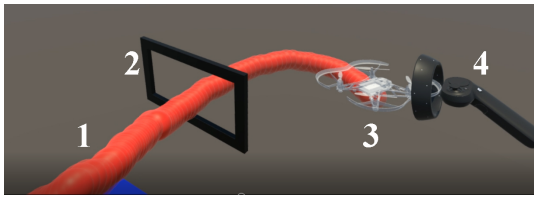

Fig. 4. Interface from the operator's viewpoint: (1) Flight-path drawn by the operator (2) Representation of the real life obstacle (3) Representation of the desired end position of the quadrotor (graspable) (4) Representation of the controller in operator's hand (5 - not in frame) Representation of real life quadrotor (not graspable)

C. Previous Research

There have been several examples of research in human-robot interfaces and control [4][5]. Noteworthy for this paper is the research of Murphy and Casper analyzing the human-robot interface of search and rescue robots. When looking into robotics used for first response they conclude that, while in theory there is a large potential in rescue robotics, if rescuers trained on the robot systems are not comfortable with the controls, the robot systems will not be used when a disaster occurs. They also argue that robotic rescue systems will never get integrated if they are too complex to be trained on [6]. This serves as the primary motivation in this work to develop an intuitive control interface that requires zero training time before use. In [7], multiple real life disaster response scenarios are described where an aerial vehicle is deployed, showing that apart from the main operator, at least 2 other trained persons are required to ensure redundant situation awareness and safe operation even in the simplest of deployments. This has been further stressed in [8] and by Pratt et al. in [9]. The second goal of this work is therefore to create an aerial robot interface that can be reliably used by only a single operator without sacrificing precision or situation awareness in any given situation.

While there is previous research on how to control multiple MAVs at the same time as one formation and alternative control methods (e.g. [10] by using a haptic device), to our knowledge there is no example where a single human operator controls multiple MAVs online, at the same time, individually, with different flight tasks. While formation flight is not the topic in this work, it could greatly increase the number of MAVs that are flown at the same time, since multiple formations could be controlled with the proposed method at the same time.

Research similar to our work can be found in the work of Thomason John et al. [11] or Rognon et al. [12]. While the use of virtual reality (VR) to control an aerial vehicle is also described in this research, there is no description of an intuitive, refined control method for an aerial platform in VR. The control of multiple platforms simultaneously, as well as a solution to spatial immersion for the human operator (as described in [13]) is not shown.

II. System Design

A. Concept

To significantly improve the interaction between human operator and aerial robotic vehicle, a VR embedded control method is proposed. In contrast to a first-person view, similar to sitting in a pilot seat, or an over the shoulder 3rd person view, the operator is able to freely walk around a virtual representation of the controlled device and its environment. This enables the operator not only to get a quick and intuitive understanding of the surrounding environment, but also enables the operator to clearly view and comprehend scale and dimensions of obstacles around the teleoperated device.

Instead of mapping the complex control of a multirotor system to two joysticks, an intuitive grab and place method using a VR hand controller is proposed. The buttons on the VR hand controller are mapped as seen in Fig. 3. With this method, the operator is able to interact with the controlled device in a simple and intuitive way, preventing human error due to faulty control inputs whenever possible (Fig. 4).

B. Setup

1) Hardware: To prototype this approach, small size, off-the-shelf quadrotors (Parrot Mambo) with the dimensions 18cm $\times$ 18cm $\times$ 5cm were used as teleoperated devices. The limited payload capabilities of this quadrotor do not allow for a stereo or RGB-D camera to be mounted to measure the surrounding environment and perform an implementation of simultaneous localization and mapping (SLAM). Therefore, in this work, the position of the devices and obstacles are measured with an external motion capture system consisting of 4 OptiTrack cameras and markers on the quadrotors and obstacles. The motion capture system is connected via wireless network and

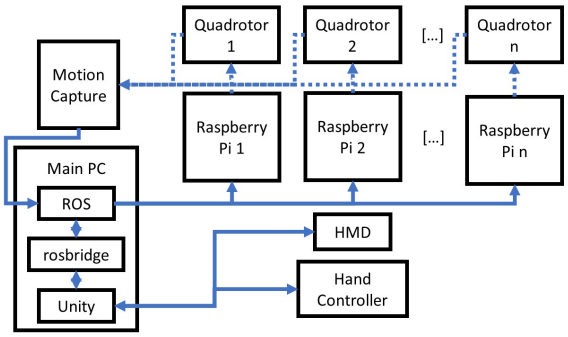

Fig. 5. Connection graph of the components used in this approach

Robot Operating System (ROS) protocol to the main computer [14]. The method was tested using a computer powerful enough to render VR environments in real-time (i5-8600k, 16GB RAM, GTX 1080).

A separate small, low performance processing board (Raspberry Pi) is setup for each quadrotor and configured to send control signals to the associated quadrotor via Bluetooth immediately after receiving new control commands via ROS protocol. The proposed method is able to handle the deployment of a large number of quadrotors simultaneously without bottlenecks in the communication pipeline. While it would be possible to control n number of quadrotors from the main computer or a single Raspberry Pi, the use of a single processing board per quadrotor guaranties the least amount of delay between receiving and sending out new control signals towards the linked quadrotor, since the control commands can be sent out simultaneously and don't need to go on the Bluetooth signal stack. It also enables a much further spatial divide between the main computer and each of the quadrotors.

For our VR head mounted display (HMD) we used the HP Windows Mixed Reality Headset and used the Windows Mixed Reality Controllers as our hand tracking devices. These devices are connected with the main computer via HDMI, USB and Bluetooth. A connection graph of the setup can be seen in Fig. 5.

2) Software: For visualization and to drive the HMD we are using the game engine Unity, running on the main computer [15]. This allows us to receive control signals from the HMD, render the virtual environment, and send an HMD ready video signal to the HMD headset without creating our own implementation from scratch. The control method was implemented as an extension for the Unity engine in `C#`.

The main computer runs the server for all ROS connected devices. It also runs ROS Bridge, a parsing server that is interpreting information received from ROS to JSON API and back. Using JSON API enables the use of ROS topic data in applications that are not native to ROS, such as Unity.

C. Implementation

A virtual space is created using routines integrated into Unity. The operator, wearing the HMD and the hand controllers is able to freely move around this virtual space. The operator is able to walk around, duck, lay down or jump to change his position while controlling the aerial robotic system, as if they were in a normal space.

Current position and alignment of the real life quadrotor P_R is measured by the motion capture system. Our implementation creates a model of the exact same scale and shape, rendered in the virtual space P_V, at an appropriate position (initially close to the operator). $P_V = P_R$ is a 6 line vector $[x_v, y_v, z_v, yaw_v, pitch_v, roll_v]^T$.

After the system recognizes a quadrotor in the start/landing zone, an additional, transparent representation of the quadrotor is spawned in the virtual space. The operator is able to pick up this transparent model with any hand controller and intuitively move it around in 3D space, just like a real world object. Once released, the transparent model will continue to float at the position where it was released.

The operator is able to start and land the quadrotor with the click of a button. The desired quadrotor for takeoff is selected by checking the current gaze of the operator, selecting the most center quadrotor in the current field of view of the operator. Take off, control $C = [x_c, y_c, z_c, yaw_c]^T$ and landing signals are handled by a customized control program running in Python on the remote Raspberry Pi platforms.

Once the operator starts a quadrotor, the color-coded, transparent representation of this quadrotor can be picked up by tightly gripping the hand controller and putting it in a desired position and alignment in 3D space P_D. Since pitch and roll position are not directly controlled when handling a multirotor in "attitude hold" mode, pitch and roll are ignored when making P_D a 4 line vector $[x_d, y_d, z_d, yaw_d]^T$.

Once the operator releases the transparent quadrotor at a desired position and presses the trigger of the hand controller that last touched it, the control signals are calculated for that system with

$$P_D(t) = P'_V(t) \tag{1}$$

$$P_E(t) = P_D(t) - P_C(t) \tag{2}$$

$$C(t) = K_p\ P_E(t) + K_i \int_0^t P_E(t')\ dt' + K_d \frac{dP_E(t)}{dt}, \tag{3}$$

with P'_V being a 4 line vector created from P_V without its $pitch_v$ and $roll_v$ parameter and K_p, K_i and K_d being tuned parameters fit for the PID controller used.

The control signal is calculated for each of the quadrotors in use by the main computer at 120Hz (the maximum frequency at which the motion capture system can supply new positional data) and published to each of the corresponding Raspberry Pi platforms every 10Hz (the maximum frequency the Parrot Mambo platform is able to receive the generated low level Bluetooth control sequences). The worst case scenario delay between position change and receiving a new control signal is therefore 108ms when ignoring the minimal network and calculation delay (in the current setup less then 1ms).

This results in a smooth movement of the real life quadrotor to the desired position and yaw heading in 3D space as long as the operator continues to press the hand controller trigger. If the trigger is released at any time, the quadrotor begins hovering at the current position P_R.

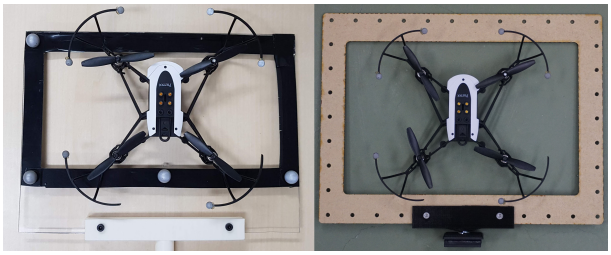

Fig. 6. Obstacle used in Experiment 1 with a hole of 23cm × 11cm (left) and obstacle used in Experiment 2 and 3 with a hole of 24cm × 16cm (right) with the quadrotor used (18cm × 18cm × 5cm) on top.

The operator is also able to draw a flight path in 3D space, that is autonomously followed by the desired quadrotor.A small sphere S_n with the counter n is spawned at the current position P'_V. Another sphere is spawned in high succession at the position of the hand controller resulting in a line-like path floating in 3D space. Instead of calculating the PID control signals towards the translucent quadrotor position P'_V, PID control signals towards the position P_{S_n} of the oldest spawned sphere are calculated. When the euclidean distance $D_{S_nV'}$ between P_{S_n} and P'_V without its yaw value:

$$D_{S_nV'} = \sqrt{(x_{S_n} - x_v)^2 + (y_{S_n} - y_v)^2 + (z_{S_n} - z_v)^2}, \quad (4)$$

is smaller than a certain threshold, the oldest sphere is despawned and the loop repeats until there are no spheres left. This "draw mode" results in a smooth movement along a line floating in 3D space.

While grabbing and also in draw mode, the system checks if any objects in the virtual space enter a bounding box around the quadrotor model of arbitrary size (in our case 18cm × 18cm × 15cm). If this box intersects with another object located at P_O at the end of each positional frame received (120Hz), a new, corrected position P'_{Vc} or P_{S_nc} is created:

$$P'_{Vc} = P'_V + \xi(P'_V \hat{-} P_O) \text{ OR} \quad (5)$$

$$P_{S_nc} = P_{S_n} + \xi(P_{S_n} \hat{-} P_O), \quad (6)$$

with ξ being an arbitrary scaling factor for the size of the normalized offset vector created. This is repeated until the collision is resolved. This simple collision avoidance method works with all projected objects within the VR-space and prevents the human operator from setting flight targets or flight paths at collision points, thus avoiding human error when setting high level movement targets for the quadrotor.

Since the scale of the real world is mapped 1:1 into VR, it sometimes proved to be bothersome for the operator to move around in our test scenarios. To mitigate that, the ability to change the human operator scale in VR has been implemented. Using the Joystick on the VR hand controller, the operator is able to dynamically change their scale, resulting in the ability to either do very wide range motions in VR with very small motions in real life or vice versa.

TABLE I

NUMBER OF TRIALS CONDUCTED IN EXP. 1 (PASSING A SINGLE OBSTACLE) AND THEIR RESULTS.

	Fails	Successes	Total Trials	Success Rate
Gamepad	15	5	20	25%
VR	2	18	20	90%
VR with CA	0	20	20	100%

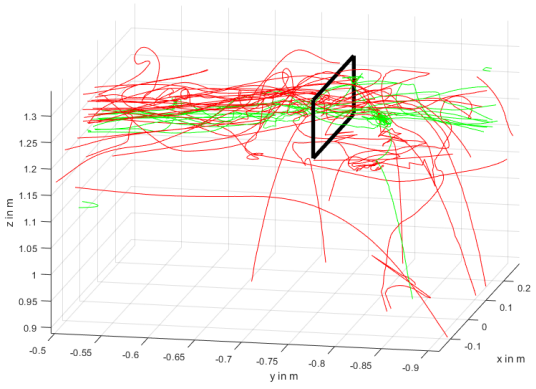

Fig. 7. Flight paths (flight direction is left to right) in Experiment 1. While trials with a traditional controller (lines in red) often resulted in a crash or entanglement with the obstacle (black), the VR control method (lines in green) mostly ensured safe maneuvering through the hole (see Table I).

III. EXPERIMENTS

A. Setup

A testing space was prepared to show the feasibility of the method presented in this work in comparison to a traditional control method for a quadrotor, namely a standard four analog axis controller in the form of a gamepad. A single obstacle with a rectangular hole of 23cm × 11cm for Experiment 1 is created from acrylic and four obstacles with a rectangular hole of 24cm × 16cm are created for Experiment 2 and 3 using laser-cut plywood and 3D printed parts (see Fig. 6). To show the impact of the collision avoidance (CA) method to mitigate human error when using the VR teleoperation, the VR method is tested with and without CA, separately.

B. Methods

In our first experiment (Exp. 1) only one quadrotor was used.

The operator tries to fly a 18cm × 18cm × 5cm quadrotor though a 23cm × 11cm hole of a single obstacle mounted on a tripod at 130cm height, using a standard 4 axis controller with the controls mapped as in Fig. 2, while standing 3 meters away from the obstacle but always being able to make visual contact. This is repeated 20 times.

The operator proceeds to try to fly the quadrotor though the hole of the same obstacle using the proposed method in virtual reality. Since the whole method is done in VR, the operator does not need to have visual contact, or need to be physically close to the obstacles or the MAVs. This is also repeated 20 times with and without implemented collision avoidance (CA).

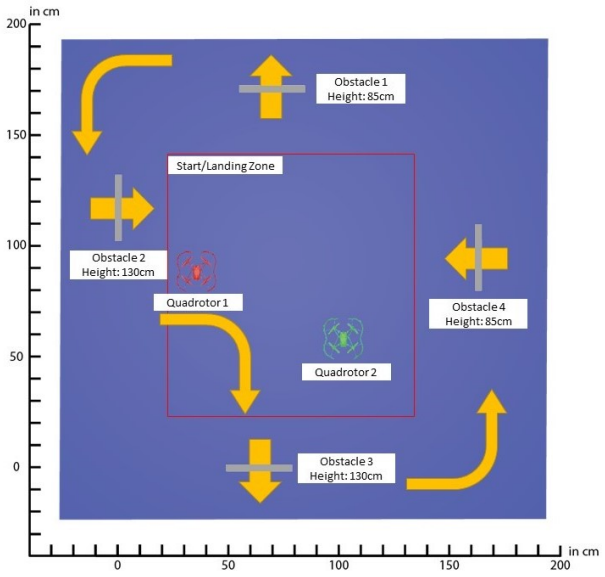

Fig. 8. Schematic of the obstacle course from the top with desired flight path (yellow) to scale. For clarity, the thickness of the obstacles (grey) has been increased by 10 times.

TABLE II

NR. AND LOCATION OF PASSES IN EXP. 2 (THOUGH OBSTACLE COURSE WITH A SINGLE QUADROTOR) & TOTAL PASS RATE (SUCCESSFULLY PASSING THOUGH ALL 4 OBSTACLES IN ONE TRIAL RUN).

	Passed though obst. 1	Passed though obst. 2	Passed though obst. 3	Passed though obst. 4	Total Trials	Total Success Rate
Gamepad	4	2	1	0	10	0%
VR	10	9	9	9	10	90%
VR with CA	10	10	10	10	10	100%

TABLE III

NR. AND LOCATION OF PASSES IN EXP. 3 (THOUGH OBSTACLE COURSE WITH MULTIPLE QUADROTORS SIMULTANEOUSLY) & TOTAL PASS RATE.

	Passed though obst. 1	Passed though obst. 2	Passed though obst. 3	Passed though obst. 4	Total Trials	Total Success Rate
Quadr. 1 & 2 in VR	10	9	9	8	10	80%
Quadr. 1 & 2 VR with CA	10	10	10	10	10	100%

A failed test run means the crash of the quadrotor or entanglement in the obstacle. Slight contact of the quadrotor and the obstacle with a following recovery were not counted as fails. The resulting 3D plot of the trials as well as the resulting table can be seen in Fig. 7 and Table I.

In the second experiment (Exp. 2), the performance of the proposed system when steering a single quadrotor though a complex obstacle course was tested against the classical control method. For that, a small obstacle course was prepared with four 24cm $\times$ 16cm obstacles. The size of the obstacles were slightly larger than in Experiment 1, since the failure rate in Experiment 1 with a traditional control proved to be too high when trying to fly though 4 obstacles in the same trial run. The position of the obstacles in 3D space was measured by the motion capture system, and since the obstacles in this work are static, they were hard-coded into the VR test routine. The four prepared obstacles should be flown through in the order and direction as shown in Fig. 8. As an additional limitation, the holes should only be flown through in a forward facing direction. This is closer to how an operator would control a quadrotor when using a traditional controller as well as a good performance test for putting a forward facing camera on the platform in the future for image processing tasks. This means not only x, y, z positions of the quadrotor need to be controlled and corrected, but also the heading (something that could be largely ignored in Experiment 1). This experiment was repeated 10 times. The results of the trials can be found in Table II.

As a last experiment (Exp. 3) the performance and feasibility of the proposed method when controlling multiple quadrotors at the same time was tested. The task given was the same as in Experiment 2, only now, both quadrotors were required to pass the current obstacle first before proceeding to the next obstacle. Depending on the situation the operator may choose to move the quadrotors either separately or simultaneously. The different quadrotors are color-coded in VR as red and green to make them easily distinguishable for the operator. A screen shot from a video showing this experiment is shown in Fig. 9 and a table with the trial results is shown in Table III.

C. Discussion

While the success rate of flying through a small size hole with a quadrotor, when using a traditional remote controller, certainly depends on the skill of the operator, the large difference in success rate when comparing the traditional control method (25% success rate in Experiment 1, 0% success rate in Experiment 2) with the VR interface approach presented (90% success rate in Experiment 1 and 2) is significant enough to show the initial feasibility of this approach. The main reason for failure when using a traditional control was the entanglement of the quadrotor frame in the obstacle or the bumping against the opening of the hole due to the inability to guess proper

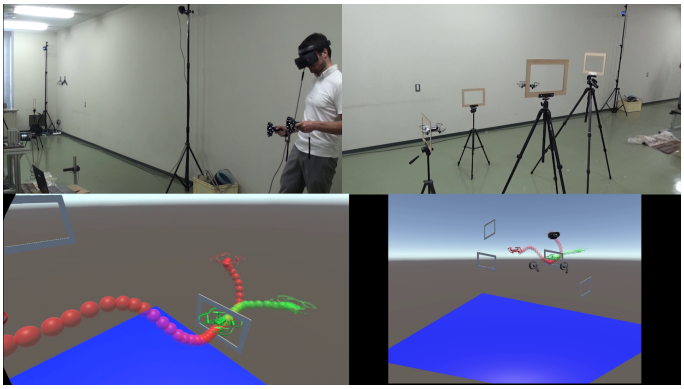

Fig. 9. Video (`https://youtu.be/LaUK_ickBeM`) showing Experiment 3. From top left to right bottom: Operator with VR headset and hand tracker; Obstacle course with quadrotors visible; Operator first person view in VR; The workspace in isometric perspective (used for visualization and debugging)

distances to the obstacle, leading to over-steering and eventual crash (see Figure 7).

Guessing distances and scale was significantly easier with the VR based teleoperation approach. This was especially apparent when controlling a quadrotor though the prepared obstacle course in Experiment 2 with a traditional remote control. Since the obstacles were angled at 90 degrees to each other, it is difficult for the human operator to visually confirm the current position of the quadrotor in relation to the obstacles. Even if constant repositioning of the point of view is possible, depending on the obstacle size and distance to the operator, position of the quadrotor in 3d space can only be approximated. This resulted in zero successful trial runs with the traditional control method through the prepared obstacle course.

In our proposed method, the human operator can quickly change their position in Experiment 1, 2 and 3 by moving in VR space and can even cross the flight paths or current real life position of the quadrotors in VR space. Hover positions and flight paths could be chosen quickly and intuitively. Due to the ability to change scale of the environment on demand, the operator was able to do large scale movements (moving the quadrotor to a new obstacle) as well as small scale movements (navigating the quadrotor through an obstacle hole) easily.

Tracking the position of multiple objects in VR, even when they are not within the field of view, is easy due to the human brain's excellent spatial memory capabilities [16]. This was demonstrated in our own study, where the operator had no difficulty managing the workload of a second quadrotor in VR space.

Failures in the VR trials can be accounted to situations where the flight path was chosen too close to the edge of the obstacle and the control algorithm could not recover the position of the quadrotor in time to avoid a crash or entanglement with the obstacle. The implementation of a simple collision avoidance method has a large positive impact on the rate of human errors as seen in Table I, II and III.

IV. Future Plans

Since in this work a small size quadrotor with limited payload capabilities was used, quadrotor and obstacle positions are measured by an external motion capture system. While a system like this would be feasible to deploy in highly controlled environments, like a factory hall, inside out sensing of the current position is desirable when deploying the system in uncontrolled environments and outdoors. Since the feasibility of the VR interface for multirotor teleoperation is shown, the use of an aerial vehicle with bigger payload capabilities is considered. This would enable the mounting of hardware needed for on-board calculation of current position as well as obstacle recognition. This new hardware can be easily added to the current setup and VR interface due to the designed scalability of the proposed system.

The proposed simple collision avoidance method was surprisingly effective in mitigating human input error in VR space. It does however, not scale well with a large number of objects in the VR space since it needs to check for collision with each object in the scene. In complex scenes, such as a projection of a point cloud, the method needs to be revisited.

There is also a need for a more objective evaluation of an aerial robotics interface. While the ease of use when using this system is immediately apparent for all operators who tried the system, all trial runs in this work were conducted by the same operator. A larger scale test trial with multiple human operators of different skill levels with an accompanying survey would be needed to evaluate the wide-scale viability of our approach.

V. Conclusion

In this paper, an alternative to the traditional two-joystick control method for teleoperated mirco aerial vehicles, especially multirotors, was presented. The presented method can be used to control multiple aerial vehicles at the same time by a single operator with high precision. A working prototype of the method as well as an initial interface was created, and the feasibility of this method was shown in several experiments.

References

[1] Various Authors, "Drone business research report 2018," *Inpress*, 2018.
[2] A. Schultz, "A timeline of nrl's autonomous systems research," 2011.
[3] D. Crane and D. Crane, *Dictionary of aeronautical terms*. Aviation Supplies & Academics Newcastle, WA, 2006.
[4] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Autonomous robots*, vol. 11, no. 1, pp. 9–18, 2001.
[5] C. Pittman and J. J. LaViola Jr, "Exploring head tracked head mounted displays for first person robot teleoperation," in *Proceedings of the 19th international conference on Intelligent User Interfaces*. ACM, 2014, pp. 323–328.
[6] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 3, pp. 367–385, 2003.
[7] R. R. Murphy, *Disaster robotics*. MIT press, 2014.
[8] R. R. Murphy, K. S. Pratt, and J. L. Burke, "Crew roles and operational protocols for rotary-wing micro-uavs in close urban environments," in *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, ser. HRI '08. ACM, 2008, pp. 73–80.
[9] K. S. Pratt, R. Murphy, S. Stover, and C. Griffin, "Conops and autonomy recommendations for vtol small unmanned aerial system based on hurricane katrina operations," *J. Field Robot.*, vol. 26, no. 8, pp. 636–650, 2009.
[10] A. Franchi, H. H. Bülthoff, and P. R. Giordano, "Distributed online leader selection in the bilateral teleoperation of multiple uavs," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 3559–3565.
[11] J. Thomason, P. Ratsamee, K. Kiyokawa, P. Kriangkomol, J. Orlosky, T. Mashita, Y. Uranishi, and H. Takemura, "Adaptive view management for drone teleoperation in complex 3d structures," in *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. ACM, 2017, pp. 419–426.
[12] C. Rognon, S. Mintchev, F. Dell'Agnola, A. Cherpillod, D. Atienza, and D. Floreano, "Flyjacket: An upper body soft exoskeleton for immersive drone control," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2362–2369, 2018.
[13] E. Adams, "Postmodernism and the three types of immersion," *Gamasutra: The Art & Business of Making Games*, vol. 9, 2004.
[14] Willow Garage, "Robot operating system," *http://www.ros.org/*, 2009.
[15] Unity Technologies, "Unity game engine," *https://unity3d.com/*, 2005.
[16] G. L. Allen, *Human spatial memory: Remembering where*. Psychology Press, 2004.