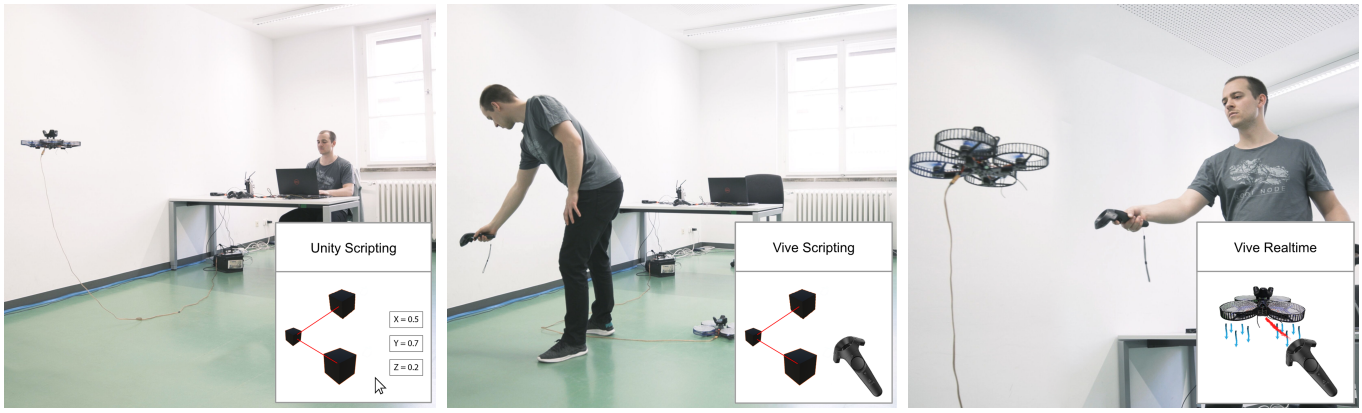
DronOS: A Flexible Open-Source Prototyping Framework for Interactive Drone Routines

Matthias Hoppe, Marinus Burger, Albrecht Schmidt, Thomas Kosch
{firstname.lastname}@ifi.lmu.de
LMU Munich, Munich, Germany



(a) (b) (c)

Figure 1: DronOS is a framework which enables novice and expert users to prototype custom drone routing and behaviour. DronOS provides the three exemplary interaction modalities (a) *Unity Scripting*, (b) *Vive Scripting*, and (c) *Vive Realtime* which enables an agile definition of flying paths.

ABSTRACT

We present DronOS, a rapid prototyping framework that can track, control, and automate drone routines. Previous research in the domain of Human-Drone Interaction relied on hardware or proprietary vendor-dependent libraries that had to be exclusively programmed for specific use cases. This forces users to stick with a drone manufacturer or model as well as limiting users in transferring their drone control logic to other drones. To overcome the aforementioned issues, our framework uses low-cost off-the-shelf hardware and applies to a variety of already available or self-crafted drones. To assess the usability of DronOS, we evaluate three drone programming modes: *Unity Scripting*, *Vive Scripting*, and *Vive Realtime*. We find that *Vive Scripting* required the least subjective workload in programming drone routines while *Unity Scripting* yielded the highest accuracy and *Vive Realtime* the least task completion time. We anticipate requirements for drone prototyping frameworks that target novice and expert users as operators.

MUM 2019, November 26–29, 2019, Pisa, Italy

ACM ISBN 978-1-4503-7624-2/19/11…$15.00
https://doi.org/10.1145/3365610.3365642

CCS CONCEPTS

• **Human-centered computing** → **User interface programming**; *User interface toolkits.*

KEYWORDS

Drones; Automation; Human-Drone Interaction

ACM Reference Format:
Matthias Hoppe, Marinus Burger, Albrecht Schmidt, Thomas Kosch. 2019. DronOS: A Flexible Open-Source Prototyping Framework for Interactive Drone Routines. In *MUM 2019: 18th International Conference on Mobile and Ubiquitous Multimedia (MUM 2019), November 26–29, 2019, Pisa, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3365610.3365642

1 INTRODUCTION

Drones have emerged as interaction devices in home and research applications. The community around the domain of Human-Drone Interaction (HDI) focuses on the use of drones as user interface [11]. Recently, a large number of various application domains took advantage of drones for several use cases. For example, drones have been used as tactile in virtual reality [17, 20], employing safety features [8, 24, 25], and supporting visually impaired people during navigation [6, 7]. Such use cases show that the flexibility of drones can be used to augment user interfaces in their functionality as well as provide context-aware in-situ interaction. Furthermore, drones can serve as levitating tangibles that enable proactive 3D interaction spaces regardless of the user positioning [19].

The recent development in HDI is fostered by the availability of drones in the consumer and commercial markets. Various research projects show that drones are a unique tool that allows objects to be quickly placed in three-dimensional space without any kind of suspension in indoor and outdoor locations [18]. Smaller hardware and the increasing demand for further use cases, such as the delivery of goods [13] and area surveillance [26], has proliferated the development of toy and custom made drones. While consumer drones can be controlled and used out-of-the-box, their usage is often limited to the intended functionalities. Thereby, extensions require direct modifications to the hardware and reverse engineering of the control logic. Furthermore, while manual drone navigation via a controller can be easily accomplished by a human operator, the automatisation of drones can be algorithmically challenging.

Several factors contribute to this issue. The programming of drones is prone to vendor-dependent libraries and communication protocols that become prone to noise if the distance between the drone and the operation unit is large. In other words, exchanging the drone entails adjustments to the current algorithmic implementation of the software control and communication unit. Furthermore, depending on the user context of the drone, the location of the drone needs to be known, which may require an expensive tracking system [17]. Currently, the automatisation of drone routines requires technical expertise to tailor them for the respective use case [18]. The aforementioned factors prevent the rapid deployment of drones for testing and research purposes. As an alternative, a Wizard of Oz approach has been used to gain initial insights into the results of employing HDI for respective use cases [5, 10]. However, specialised labour is necessary to operate the drone.

This work presents DronOS, a Unity-based drone framework that circumvents the aforementioned challenges. DronOS represents a drone routing prototyping platform (see Figure 1). DronOS uses affordable off-the-shelf hardware that is part of the commercially available HTC Vive bundle[1]. DronOS can be employed into new or existing Unity projects and uses HTC Vive Lighthouses as a tracking system (see Figure 2). The software counterpart is a framework in Unity that allows for easy programming of the drone. Programming by demonstration and manipulation of checkpoints are supported using the Unity framework. We conduct a study to showcase and compare three automated path navigation modalities using DronOS. Furthermore, we discuss the design implications that support interface designers when creating new HDI.

CONTRIBUTION STATEMENT

Our work makes three key contributions: We (1) describe DronOS, a framework that uses affordable hardware for drone navigation and automatisation. DronOS enables expert as well as novice users to orchestrate drone movement sequences through a Unity interface. Furthermore, we (2) present a study in which three route planning modalities are showcased and evaluated regarding their route planning efficiency. Finally, we (3) discuss the relevant implications and lessons learned for future frameworks that aim to simplify the programming of drone behaviours.

[1] www.vive.com - last access 2019-10-10

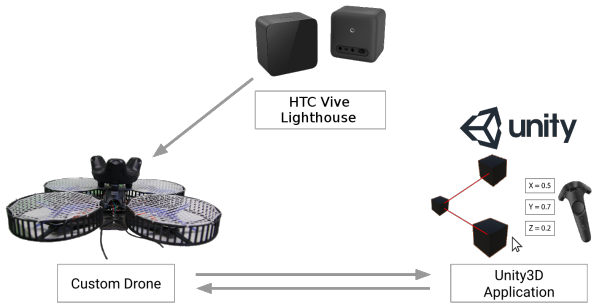


Figure 2: Key components of DronOS: The Unity-based framework DronOS provides drone routines. The radio control connection allows the usage of custom and off-the-shelf drones. The HTC Vive Lighthouse system is used as tracking system.

2 RELATED WORK

We present recent research that has tackled the challenges of drone orchestration and interaction in the following. With BitDrones [12, 28], a toolbox for drone movement sequences and orchestration is presented. Drones may represent levitating tangibles for interaction, where users can perform input or perceive output using RGB lights or displays. However, the current implementation of BitDrones relies on custom-made drones and an expensive tracking system. Cauchard et al. [9] presented Drone.io, a platform that is capable of providing a variety of gesture- and foot-based interactions. In three user studies, several interactions and interface designs were evaluated. With drones becoming more and more present, drone interaction concepts that do not disrupt daily tasks are increasing in importance. Relevant factors, such as proxemics, safety risks, and transparency have to be communicated with users to assure non-problematic interaction [31]. Abtahi et al. [3] investigated how drones serve as a proxy for objects which provide haptic interaction. They demonstrate the feasibility of their system in a virtual shopping experience and evaluate different drone affordances that contribute to drone interaction. Past research has investigated concepts related to HDI. In the following, we present recent research which uses these concepts to provide interaction with drones.

Drones extend tangible interaction within a 3D space. Knierim et al. [19] explored modalities that use the 3D space for interaction. They evaluate input and output modalities, such as pushing or being dragged by drones, in their suitability for interaction. A physical interaction space is derived which helps user interface designer to understand how paradigms influence spatial relationships between users and drones. Besides for interaction purposes, autonomously flying drones have been used to capture video material from a perspective that is otherwise difficult to reach. In XPose [22], three predefined interaction modes to take photos using drone-mounted cameras are presented. The interaction modes were designed to simplify secondary tasks, such as taking photos, while controlling a drone. While the aforementioned work uses a touch interface to interact with the functionality of drones, Kosch et al. [21] evaluates pointing gestures with and without visual feedback to control drones in a 3D space. Their findings include a trade-off between

accuracy and speed for drone positioning. Using visual feedback while performing a gesture yielded a higher accuracy compared to obviating visual feedback. However, performing gestures with no visual feedback resulted in faster task completion times.

Hoppe et al. [17] explored the use of drones to augment the haptic perception of objects in virtual reality. Their findings endorse the use of flexible drone positioning to dynamically adjust the haptic experience. However, haptic stimulation by drones may vary in their intensity and requires a realistic mapping between a perceived scene and the haptic sensation. Therefore, Abdullah et al. [1, 2] investigated how different haptic and tactile stimulation sensitivities of drones need to be adjusted to provide a realistic as possible interaction scenario. However, such close interactions with drones may elicit security issues among users. Therefore, previous research has investigated how users interact with nearby drones [4] where interactions via gestures, touch, and sound, as well as a multimodal combination, was confirmed as comfortable by the participants.

Drones have shown to be beneficial as navigation modality for visually impaired people. Avila et al. [6] demonstrated how the noise of drones can be used for navigation. The produced sound is envisioned as a promising method for navigation. Furthermore, drones can carry objects which are intended to provide navigation cues for visually impaired people. For example, a leash carried by a drone [7] has been used for guidance. Less time was required as well as fewer navigation errors were made using acoustic or leashed drones compared to traditional audio instructions.

The implementation of custom drone control loops is often restricted to the programming interfaces which are provided by the manufacturers. Furthermore, proprietary radio interfaces, such as modified Bluetooth stacks or encrypted communication channels, make it difficult for novice users to program a custom drone interface. We close this gap by proposing DronOS, a prototyping framework capable of automating drone flight processes. Drone flight behaviors can be programmed using a Unity interface that provides functionalities suitable for novice and experts. DronOS uses standardised software toolkits and off-the-shelf hardware to establish a communication between the drone and intended user action.

3 DRONOS: A DRONE NAVIGATION FRAMEWORK

Previous work informed about the relevant requirements of drones for user interaction. We present the concept, implementation details, and user interface of DronOS in the following.

3.1 Concept

Autonomous drone routing has been a challenging topic in various contexts [30]. Drones have to provide a reliable connection regarding tracking as well as remote control purposes, employ safety-related features, and communicate the influence of adjusted drone parameters with users. To accomplish these requirements for use cases in-home and research, drones need to provide a user-friendly programming interface, stable communication protocol, and rapid prototype features. By using standardised software packages, a Unity-based communication interface, and available off-the-shelf hardware DronOS unifies these aspects in one framework. Thereby, DronOS can be used with any commercially available or custom-made drones. DronOS comes with three routing planning modalities: *Unity Scripting*, *Vive Scripting*, and *Vive Realtime*. In the following, we present technical details regarding the realisation of DronOS and elucidate the routing modalities.

3.2 Tracking

DronOS uses HTC's Lighthouse[2] tracking technology which is available within a regular HTC Vive setup. The HTC Lighthouses offer a simple calibration procedure in indoor environments. The HTC Lighthouses use infrared lights to determine where objects are located. The current iteration of the Lighthouse system covers a tracking space of up to 10×10 meters.

Conversely, drones have to be equipped with a Vive Tracker[3] to be tracked. With the Lighthouse tracking system, the drone only needs to be outfitted with a Vive Tracker (see Figure 3). Alternatively, one of the many home-brew Vive Tracker alternatives can be used, which may be available in lighter and smaller sizes. The HTC Lighthouse tracking system only needs a few minutes of set up and calibration. This enables easy setup, low budget tracking for automatisation, and space-saving setup as the HTC Lighthouse boxes can either be mounted on a wall, ceiling, or on a tripod. We use a 3D printed retainer to mount the Vive Tracker on the drone.

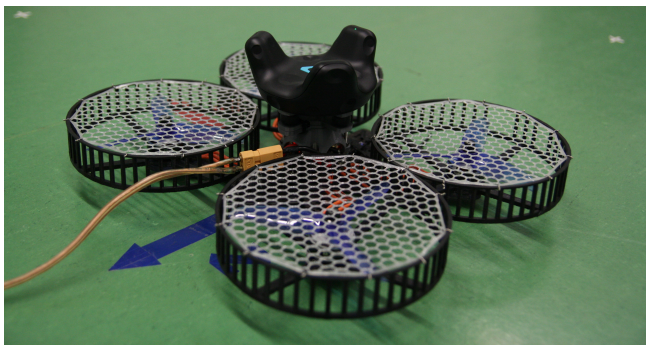


Figure 3: Drone Prototype with a 3D-printed customisable frame, radio controlled communication interface, HTC Vive Tracker, and safety nets.

3.3 Unity

Unity[4] is a 3D game engine that comes with a programming interface. We use Unity to check the position of the drone and calculate commands according to the planned drone routing and the HTC Lighthouses. To record the position of the Vive Tracker mounted on the drone we use the SteamVR Unity plugin. This allows the integration of the Vive Tracker mounted on the drone and registration of input commands that are used for programming the pathing via programming by demonstration.

[2] www.vive.com/eu/accessory/base-station - last access 2019-10-10
[3] www.vive.com/eu/vive-tracker - last access 2019-10-10
[4] www.unity3d.com/unity - last access 2019-10-10

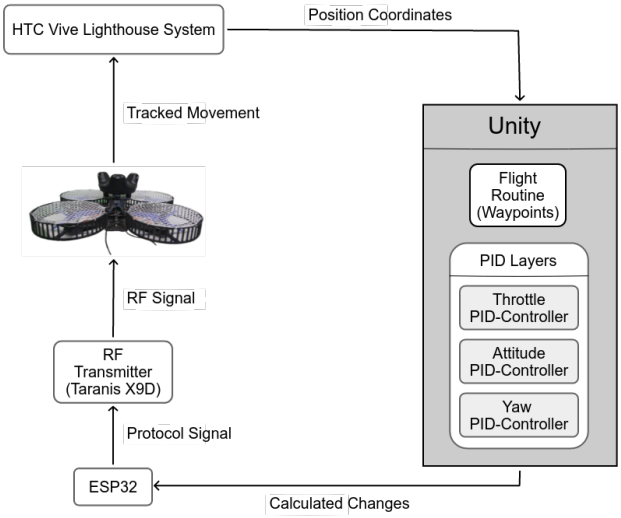


Figure 4: Control loop of the framework. DronOS controls the PID parameters and uses an ESP32 to transmit flight commands via radio control.

The Unity programming interfaces makes the planning and modification of flight routines more accessible for a non-expert. DronOS makes use of this programming interface. Through Unity, it allows programming and scripting of drone flight paths and behaviours without any prior knowledge of coding, as it can be used in a fully visual fashion. While prior Unity knowledge helps to use the framework, it is not necessary. DronOS comes with implementations of the three drone programming modes *Unity Scripting*, *Vive Scripting*, and *Vive Realtime* on which we elaborate in the following.

3.3.1 Unity Scripting. This mode utilises the Unity user interface for the detailed definition of routines. New waypoints can be created by drag-and-drop of so-called "Waypoint Objects" that represent cubes as game objects (see Figure 5a). The user can change the position by dragging or entering the coordinates of the game objects and if desired. Settings such as dwelling times (i.e., the drone remains the same position) can be changed by modifying values the properties of each node. General settings regarding the drones PID can be adjusted as well. The available tracking area is presented by a border in the Unity scene, as the SteamVR Plugin is used for the tracking components. Positions and distances of waypoints can either be set by moving the waypoint along X-, Y-, and Z-axis or entering the coordinates directly. Conversely, this requires the user to measure the distances manually.

To set a waypoint in a specific real-world location, the coordinates have to be entered to the zero point of the tracking space. For this method measuring the real space is required to get the correct coordinates This approach is similar to most current systems used in research. This is the most detailed mode that requires more work in creating a flight routine, but also offer the most options for single waypoints. Further, this mode can be used to tweak flight routines that are created in other modes. Users then confirm their adjustment to start the drone.

3.3.2 Vive Scripting. A programming-by-demonstration approach that creates flight routines by marking positions in the real world. The user creates the next waypoint in a routine by positioning a handheld Vive controller[5] at the desired location (see Figure 5b). The Vive controller employs the same technology as the Vive Trackers and can thus be used with the Lighthouse tracking technology. The position is logged by clicking a button on the controller. This approach is a fast and easy way of prototyping a flight routine and setting up waypoints in real space without the need for measuring distances and entering coordinates manually. If required, the flight routine can be tweaked by going to the desktop and editing waypoint nodes in the Unity Scripting mode.

3.3.3 Vive Realtime. Similar to the work presented by Kosch et al. [21], the user controls the drone directly by pointing with a Vive controller to the final target. The drones fly then, in real-time, to the pointed position (see Figure 5c). The steering of the drone is easier as the drone is following the user's hand movements. This gives the user a feeling for the drone and its impact on the world, as it directly follows the user's input. This live-performance of the flight path can be recorded and could later be played back by ghosting the user's movements.

3.4 PID Controller

To accomplish autonomous flight, a control system had to be designed that reacts appropriately to the current tracked position and target positions. This was implemented as a set of layered Proportional-Integral-Derivative controllers (PID controllers) which is derived from the PID controller architecture by Luis and Ny [23]. A PID controller calculates the discrepancy of the current position and the next waypoint, and then applies a correction based on proportional, integral, and derivative characteristics. A preset of PID values are already available, as setting these requires some experience and expertise, but can be tweaked if desired. When a drone flies a path, the impact of the changes is directly applied to the drone flight process. As HTC Vive Lighthouses and the Vive Trackers update with 90 Hertz, the rate of the PID controller was set to 90 Hertz as well.

3.5 Communication

For the drone being able to execute the commands, they need to be transmitted from the Unity programming interface. Therefore, the calculated commands need to be sent over a transmitter to the drone. In this setup we used a FrSky Taranis X9D[6] remote as the transmitter. The remote is usually used as an input device for a human user, but in this case, the input will be automated by the Unity interface. The Taranis X9D transmitter runs OpenTX[7], an open-source firmware for RC radio transmitters. OpenTX is rich in features, highly configurable and can be incorporated in any other controller that supports OpenTX. The communication between computer and transmitter was realised through a micro-controller that generates the required protocol signal to the radio frequency

[5] www.vive.com/eu/accessory/controller - last access 2019-10-10
[6] www.frsky-rc.com/product/taranis-x9d-plus-2 - last access 2019-10-10
[7] www.open-tx.org - last access 2019-10-10

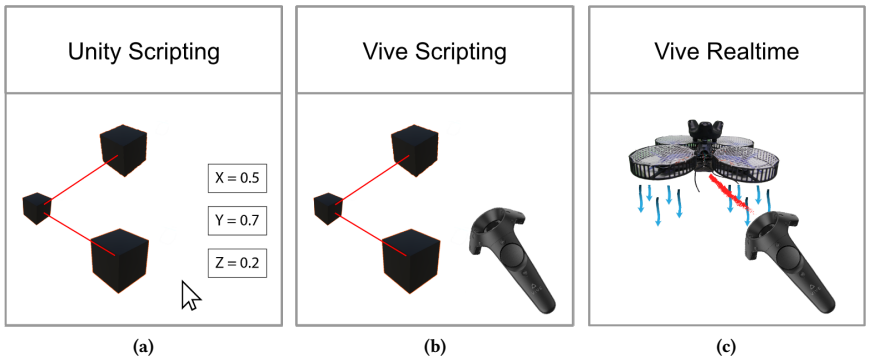


Figure 5: Three drone routine modalities implemented in DronOS. (a): User defines the routing path directly in Unity. (b): Defining a flight path with HTC Vive controllers. (c): Controlling a drone with real-time feedback using pointing gestures and an HTC Vive controller as a remote pointing device.

(RF) transmitter. We chose the ESP32[8] microcontroller as our platform for translating the signals for the transmitter. The ESP32 has several dedicated hardware peripherals such as RF modules, wireless LAN or Bluetooth. The ESP32 development boards come with a hardware peripheral remote control module (RMT) driver, which is capable of generating very accurate high-frequency signals. A Pulse Position Modulation (PPM) protocol was implemented, which together with the RMT driver module of the ESP32, leads to a safe connection combined with the transmitter. The ESP32 generates a PPM signal from the serial input and after level converting the logic level 3.3V to 5V this signal is forwarded to the trainer port of the transmitter. The transmitter then operates the external RF module that was bound to the RC receiver on the drone. An overview of the communication can be found in Figure 4.

3.6 Drone Hardware

We use a custom drone to evaluate DronOS. However, it is possible to use every commercially available drone in combination with DronOS. For the custom drone, we decided to use components (e.g., materials, motors, propellers, etc.) that minimise the payload and maximise the flight time. This gives researchers the possibility to attach additional hardware, such as sensors or cameras, to the drone.

The drone employs a Matek F405-CTR[9] flight controller running the Betaflight[10] flight controller firmware. The drone incorporated four motors from the type T-Motor F40 III 2600KV which can lift 1.398 kilograms per motor according to laboratory testing environments [27]. The same motors have been used in previous related research projects [15]. The remaining parts, such as retainers for motors and the flight controller, were 3D printed. Furthermore, 3D-printed protective nets were placed over the rotors in case users go in direct contact with the drone. The prototype is compatible with both external power supplies and commonly used aircraft-mounted batteries.

[8]www.espressif.com/en/products/hardware/esp32/overview - last access 2019-10-10
[9]www.mateksys.com/?portfolio=f405-ctr - last access 2019-10-10
[10]https://betaflight.com/ - last access 2019-10-10

4 EVALUATION

We evaluate DronOS regarding its usability with the three aforementioned drone routing programming modes *Unity Scripting*, *Vive Scripting*, and *Vive Realtime* (see Figure 5). We elaborate the study details in the following.

4.1 Methodology

We employ a within-subject design that considers the programming mode as the only independent variable. Each participant had to program three routes of six waypoints and the same difficulty in a dedicated room with each modality. This results in a total number of nine drone programming trials per participant. The interaction area of the drone routes was 3×3 meters. We counterbalanced the sequence of modalities and tasks for the participants to preclude training effects. Before starting with each programming modality, participants were made familiar with the respective condition.

We measure the Task Completion Time (TCT) for each route participants programmed and assess the workload via a raw NASA-TLX questionnaire [14] after each trial. Furthermore, we measure the Root-Mean-Square-Error (RMSE) between the final position and the defined drone positions. Past research used the RMSE as a reliable metric to evaluate the accuracy of flight paths [16, 23]. The PID-controller values were not changed throughout the experiment to retrieve comparable results. The duration of the whole study was around 2 hours per participant, including the time participants took give informed consent, provide demographic data, and to get familiar with the different modalities.

4.2 Participants

We evaluated them in a user study with 12 participants with an average age of $M = 27.3$ ($SD = 2.9$).

Twelve participants (eight male, four female) with an average age of $M = 27.3$ ($SD = 2.9$) years took part in the study. Asking after the experience with drones revealed that eight participants had no experience with drones while three participants had moderate and one participant professional experience with drones.

4.3 Results

We submit the TCTs, raw NASA-TLX scores, and RMSE errors to a repeated-measures ANOVA to investigate for statistical differences within the programming modalities. Post-hoc tests are performed where significant main effects are found. Effect sizes are reported using Cohen's d.

4.3.1 Task Completion Time. We found a statistical significant main effect between the different programming modalities ($F(2, 22) = 15.94$, $p < .001$). A Bonferroni post-hoc test revealed a statistically significant difference on TCT between *Unity Scripting* and *Vive Scripting* ($p = .006$, $d = 1.168$) and *Vive Realtime* ($p < .001$, $d = 1.667$). Averaged in seconds, *Unity Scripting* required the most time ($M = 1376.2$, $SD = 91.88$), followed by *Vive Scripting* ($M = 928.1$, $SD = 89.62$) and *Vive Realtime* ($M = 817.3$, $SD = 56.76$).

4.3.2 Subjective Workload. A statistically significant main effect was found in the raw NASA-TLX scores ($F(2, 22) = 14.87$, $p < .001$). A Bonferroni post-hoc test revealed a significant effect between *Unity Scripting* and *Vive Realtime* ($p = 0.01$, $d = -1.075$) as well as *Vive Realtime* and *Vive Scripting* ($p < 0.001$, $d = 1.967$). Thereby, *Vive Scripting* elicited the least workload ($M = 31.25$, $SD = 8.9$), followed by Unity Scripting ($M = 39.42$, $SD = 19.59$) and Vive Realtime ($M = 55.1$, $SD = 13.95$).

4.3.3 Root-Mean-Square-Error. Applying an ANOVA to the RMSE does not result in a statistical significant main effect ($p > .05$). Averaged in centimeters and along the three axes X, Y, and Z, *Unity Scripting* yielded a higher accuracy ($M = 12.27$, $SD = 1.33$) compared to Vive Scripting ($M = 12.47$, $SD = 1.48$) and *Vive Realtime* ($M = 13.4$, $SD = 1.44$).

5 DISCUSSION

We conducted a study to evaluate the usability of DronOS. We discuss the implications of our results and potential requirements for future drone programming frameworks.

5.1 The Right Modality for the Right Job

While all of our participants, regardless of their expertise, were able to define the given flying routes, significant differences in TCTs and subjectively perceived workload were found for different programming modalities. If time is a critical factor that should be minimised, a real-time programming modality should be used, similar to our implementation of the *Vive Realtime* programming mode. However, this comes at the cost of subjective workload as indicated by increased NASA-TLX scores as well as the need for a human operator. In contrast, if workload should be minimized, a drone framework should offer a programming-by-demonstration approach similar to *Vive Scripting*. While our results approve that *Vive Scripting* elicits the least workload, it requires more time compared to *Vive Realtime*. Finally, although not statistically significant, *Unity Scripting* reveals a slightly higher accuracy compared to *Vive Scripting* and *Vive Realtime*. The future design of drone frameworks can optimize the respective programming modality that should optimize time, workload, or accuracy.

5.2 Technical Design for HDI

We present an exemplary design of a drone for HDI research. Based on past experiences in drone research, difficulties arise when programming drones to follow a specific behaviour. Furthermore, safety-related features need to be embedded on the drone to achieve a high user affordance. The presented framework, controller software, and drone were designed to follow security standards and allow the extension of the framework with minimal knowledge in programming. DronOS uses a radio controller protocol to communicate commands with the drone. We have chosen a radio controller since it provides a more reliable connection compared to Bluetooth which is incorporated in many commercially available devices [29].

6 LIMITATIONS

We are aware that our study is prone to certain limitations. The assessment of the drone expertise is subjective and might have been reported different by each participant. Furthermore, participants who were proficient with the HTC Vive setup could have achieved better performance compared to non-proficient users. Through the micro-vibrations of the drone, the tracker could temporally lose its tracking connection to the HTC Lighthouse tracking system. A potential solution might be the incorporation of multiple trackers. However, the evaluation of tracking efficiency was not the scope of the presented study.

7 CONCLUSION AND FUTURE WORK

In this work, we present DronOS, a Unity-based prototyping framework that enables us to automatise drone routines. DronOS provides an interface to define routines, such as movements between user-defined points and dwell times. We present the system architecture as well as communication between our framework and drone. Thereby, we use standardised radio control protocols for communication. This is complemented by evaluating the usability of the framework using three exemplary programming modes: *Unity Scripting*, *Vive Scripting*, and *Vive Realtime*. We find that all participants were able to successfully define the given drone routines with all programming modalities. However, our study reveals a trade-off in terms of task completion time, workload, and accuracy for each modality. We discuss, which of the programming modalities are recommended for the respective critical factor. Furthermore, we elaborate on drone control design for Human-Drone Interaction. Overall, we envision DronOS as a tool for rapid prototyping which can be used by researchers and users to evaluate novel drone interfaces.

In future work, we plan to investigate multimodal programming of the presented drone routing mechanisms. For example, a combination of *Vive Scripting* with *Vive Realtime* can be used to correct predefined paths in real-time. Furthermore, we envision DronOS as a community-driven open-source framework. To foster the development and research in this direction, we publish the software of DronOS, the flight controller, documentation, and the 3D printable files of the drone, as well as the Vive tracker, mount on a publicly available repository[11].

[11] www.github.com/HCUM/dronos - last access 2019-10-10

REFERENCES

[1] Muhammad Abdullah, Minji Kim, Waseem Hassan, Yoshihiro Kuroda, and Seokhee Jeon. 2017. HapticDrone: An Encountered-Type Kinesthetic Haptic Interface with Controllable Force Feedback: Initial Example for 1D Haptic Feedback. In *Adjunct Publication of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM, New York, NY, USA, 115–117. https://doi.org/10.1145/3131785.3131821

[2] M. Abdullah, M. Kim, W. Hassan, Y. Kuroda, and S. Jeon. 2018. HapticDrone: An encountered-type kinesthetic haptic interface with controllable force feedback: Example of stiffness and weight rendering. In *2018 IEEE Haptics Symposium (HAPTICS)*. 334–339. https://doi.org/10.1109/HAPTICS.2018.8357197

[3] Parastoo Abtahi, Benoit Landry, Jackie (Junrui) Yang, Marco Pavone, Sean Follmer, and James A. Landay. 2019. Beyond The Force: Using Quadcopters to Appropriate Objects and the Environment for Haptics in Virtual Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 359, 13 pages. https://doi.org/10.1145/3290605.3300589

[4] Parastoo Abtahi, David Y. Zhao, Jane L. E., and James A. Landay. 2017. Drone Near Me: Exploring Touch-Based Human-Drone Interaction. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 34 (Sept. 2017), 8 pages. https://doi.org/10.1145/3130899

[5] Majed Al Zayer, Sam Tregillus, Jiwan Bhandari, Dave Feil-Seifer, and Eelke Folmer. 2016. Exploring the Use of a Drone to Guide Blind Runners. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '16)*. ACM, New York, NY, USA, 263–264. https://doi.org/10.1145/2982142.2982204

[6] Mauro Avila, Markus Funk, and Niels Henze. 2015. DroneNavigator: Using Drones for Navigating Visually Impaired Persons. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS '15)*. ACM, New York, NY, USA, 327–328. https://doi.org/10.1145/2700648.2811362

[7] Mauro Avila Soto, Markus Funk, Matthias Hoppe, Robin Boldt, Katrin Wolf, and Niels Henze. 2017. DroneNavigator: Using Leashed and Free-Floating Quadcopters to Navigate Visually Impaired Travelers. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '17)*. ACM, New York, NY, USA, 300–304. https://doi.org/10.1145/3132525.3132556

[8] Anke M. Brock, Julia Chatain, Michelle Park, Tommy Fang, Martin Hachet, James A. Landay, and Jessica R. Cauchard. 2018. FlyMap: Interacting with Maps Projected from a Drone. In *Proceedings of the 7th ACM International Symposium on Pervasive Displays (PerDis '18)*. ACM, New York, NY, USA, Article 13, 9 pages. https://doi.org/10.1145/3205873.3205877

[9] J. R. Cauchard, A. Tamkin, C. Y. Wang, L. Vink, M. Park, T. Fang, and J. A. Landay. 2019. Drone.io: A Gestural and Visual Interface for Human-Drone Interaction. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 153–162. https://doi.org/10.1109/HRI.2019.8673011

[10] Jane L. E, Ilene L. E, James A. Landay, and Jessica R. Cauchard. 2017. Drone & Wo: Cultural Influences on Human-Drone Interaction Techniques. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6794–6799. https://doi.org/10.1145/3025453.3025755

[11] Markus Funk. 2018. Human-drone interaction: let's get ready for flying user interfaces! *Interactions* 25, 3 (2018), 78–81.

[12] Antonio Gomes, Calvin Rubens, Sean Braley, and Roel Vertegaal. 2016. BitDrones: Towards Using 3D Nanocopter Displays As Interactive Self-Levitating Programmable Matter. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 770–780. https://doi.org/10.1145/2858036.2858519

[13] M. R. Haque, M. Muhammad, D. Swarnaker, and M. Arifuzzaman. 2014. Autonomous quadcopter for product home delivery. In *2014 International Conference on Electrical Engineering and Information Communication Technology*. 1–5. https://doi.org/10.1109/ICEEICT.2014.6919154

[14] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139 – 183. https://doi.org/10.1016/S0166-4115(08)62386-9

[15] Seongkook Heo, Christina Chung, Geehyuk Lee, and Daniel Wigdor. 2018. Thor's hammer: An ungrounded force feedback device utilizing propeller-induced propulsive force. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 525.

[16] Gabriel Hoffmann, Steven Waslander, and Claire Tomlin. [n. d.]. *Quadrotor Helicopter Trajectory Tracking Control*. https://doi.org/10.2514/6.2008-7410 arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2008-7410

[17] Matthias Hoppe, Pascal Knierim, Thomas Kosch, Markus Funk, Lauren Futami, Stefan Schneegass, Niels Henze, Albrecht Schmidt, and Tonja Machulla. 2018. VRHapticDrones: Providing Haptics in Virtual Reality Through Quadcopters. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia (MUM 2018)*. ACM, New York, NY, USA, 7–18. https://doi.org/10.1145/3282894.3282898

[18] Matthias Hoppe, Thomas Kosch, Pascal Knierim, Markus Funk, and Albrecht Schmidt. 2019. Are Drones Ready for Takeoff? Reflecting on Challenges and Opportunities in Human-Drone Interfaces.

[19] Pascal Knierim, Thomas Kosch, Alexander Achberger, and Markus Funk. 2018. Flyables: Exploring 3D Interaction Spaces for Levitating Tangibles. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '18)*. ACM, New York, NY, USA, 329–336. https://doi.org/10.1145/3173225.3173273

[20] Pascal Knierim, Thomas Kosch, Valentin Schwind, Markus Funk, Francisco Kiss, Stefan Schneegass, and Niels Henze. 2017. Tactile Drones - Providing Immersive Tactile Feedback in Virtual Reality Through Quadcopters. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, New York, NY, USA, 433–436. https://doi.org/10.1145/3027063.3050426

[21] Thomas Kosch, Markus Funk, Daniel Vietz, Marc Weise, Tamara Müller, and Albrecht Schmidt. 2018. DroneCTRL: A Tangible Remote Input Control for Quadcopters. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings (UIST '18 Adjunct)*. ACM, New York, NY, USA, 120–122. https://doi.org/10.1145/3266037.3266121

[22] Ziquan Lan, Mohit Shridhar, David Hsu, and Shengdong Zhao. 2017. XPose: Reinventing User Interaction with Flying Cameras.. In *Robotics: Science and Systems*.

[23] Carlos Luis and Jérôme Le Ny. 2016. Design of a trajectory tracking controller for a nanoquadcopter. *arXiv preprint arXiv:1608.05786* (2016).

[24] Sven Mayer, Pascal Knierim, PW Wozniak, and Markus Funk. 2017. How drones can support backcountry activities. In *Proceedings of the 2017 natureCHI workshop, in conjunction with ACM mobileHCI*, Vol. 17. 6.

[25] Sven Mayer, Lars Lischke, and Pawel W. Woźniak. 2019. Drones for Search and Rescue. In *1st International Workshop on Human-Drone Interaction*. Ecole Nationale de l'Aviation Civile [ENAC], Glasgow, United Kingdom. https://hal.archives-ouvertes.fr/hal-02128385

[26] Gregory S McNeal. 2014. Drones and aerial surveillance: Considerations for legislators. *Brookings Institution: The Robots Are Coming: The Project on Civilian Robotics* (2014).

[27] T motor f40 iii 2306 2600kv. 2019. racedayquads. www.racedayquads.com/collections/multigp-spec-racing-class/products/t-motor-f40-iii-2306-2400kv-v3

[28] Calvin Rubens, Sean Braley, Antonio Gomes, Daniel Goc, Xujing Zhang, Juan Pablo Carrascal, and Roel Vertegaal. 2015. Bitdrones: Towards levitating programmable matter using interactive 3d quadcopter displays. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 57–58.

[29] Rahul C. Shah, Lama Nachman, and Chieh-yih Wan. 2008. On the Performance of Bluetooth and IEEE 802.15.4 Radios in a Body Area Network. In *Proceedings of the ICST 3rd International Conference on Body Area Networks (BodyNets '08)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, Article 25, 9 pages. http://dl.acm.org/citation.cfm?id=1460257.1460291

[30] E. Vattapparamban, İ. Güvenç, A. İ. Yurekli, K. Akkaya, and S. Uluağaç. 2016. Drones for smart cities: Issues in cybersecurity, privacy, and public safety. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*. 216–221. https://doi.org/10.1109/IWCMC.2016.7577060

[31] A. Wojciechowska, J. Frey, S. Sass, R. Shafir, and J. R. Cauchard. 2019. Collocated Human-Drone Interaction: Methodology and Approach Strategy. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 172–181. https://doi.org/10.1109/HRI.2019.8673127