POD: A Smartphone That Flies

Guojun Chen
guojun.chen@yale.edu
Yale University

Noah Weiner
noah.weiner@yale.edu
Yale University

Lin Zhong
lin.zhong@yale.edu
Yale University

ABSTRACT

We present POD, a smartphone that flies, as a new way to achieve hands-free, eyes-up mobile computing. Unlike existing drone-carried user interfaces, POD features a smartphone-sized display and the computing and sensing power of a modern smartphone. We share our experience in prototyping POD, discuss the technical challenges facing it, and describe early results toward addressing them.

CCS CONCEPTS

• **Computer systems organization → Robotic autonomy**.

KEYWORDS

Content stabilization, Smartphone, Human following

1 INTRODUCTION

Our driving vision is *hands-free, eyes-up* mobile computing. That is, mobile users are able to interact with a computer without holding a device in hand and looking down at it. The key to this vision is a user interface (both input and output) that does not require a human hand. Wearable devices, such as Google Glass, are perhaps the most studied implementation of such hands-free user interfaces.

This paper explores an alternative idea: a user interface carried by a small drone. Compared to wearable user interfaces, drone-carried interfaces do not impose gadgets on the user's head or eyes. More importantly, they can go beyond the physical reach of the human body, empowering interesting use cases.

While there is a growing literature about human-drone interaction [13, 14, 17, 32], this work envisions a drone-carried interface as rich as that of modern smartphones, called POD. Unlike previously studied drone-carried user interfaces, POD features a small (potentially foldable) display and allows bidirectional interactions with mobile users. We imagine that POD would rest on the user's shoulder most of the time and fly out occasionally to interact with the user only on demand. We were inspired in part by characters from popular culture, e.g., Jack the Monkey, a pet of Captain Barbossa, EVE from *WALL-E*, the Decepticon spies of Soundwave from *Transformers*, as well as Tinker Bell from *Peter Pan*.

Compared to wearable glasses and other drone-carried interfaces, POD faces its own unique challenges that have not been addressed by the human-drone interaction community. First, like all flying objects, drones are not steady, especially small drones. This challenges the usability of an onboard display, especially if the drone needs to maintain a safe distance of one to two feet from the user's eyes. Second, drones are noisy. Even the quietest drone operating at one or two feet away from the user produces noise at 70 dBA, which is equivalent to being on a busy street or near a vacuum cleaner [8]. Unlike other applications, our drone will always keep a close distance with the user, which makes it more challenging. Finally, because we intend POD to serve the user on the go, it must decide between following the user or hovering in place when the user moves around. This challenge represents a very specialized case of human-drone interaction and requires new investigation.

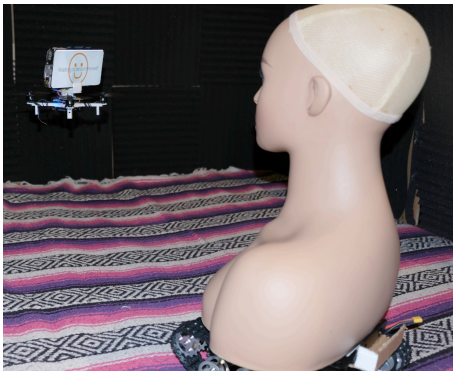

Figure 1: POD (top left corner) interacting with our test mannequin mounted on a Wi-Fi-controlled robotic chassis.

To address these challenges and experiment with POD, we have implemented a prototype. As shown in Figure 1, our prototype is based on a mechanical, electrical and computational partnership between a micro-controller-based drone and a smartphone (Google Pixel 4). It uses computer vision to track and follow its user, harnesses the phone's sensors to improve the usability of its display, and has a flight time of four minutes.

This paper describes the POD prototype and shares our experience in building it (§4). It presents our early results (§5) in addressing the challenges facing POD (§3) and discusses interesting, novel use cases for it (§2).

2 THE CASE FOR POD

We first describe several compelling applications of POD, a smartphone that flies. Because POD carries cameras and microphones, it can fill roles intended for existing cinematographic drones. We ignore these use cases and instead focus on those uniquely enabled by POD's display. We envision that POD will share much of the hardware and software of smartphones. Our prototype uses a Google Pixel 4, allowing it to tap into the mature ecosystem of Android, which facilitates app development and enables "retrofitting" existing

Dronet '21, June 24, 2021, Virtual, WI, USA

ACM ISBN 978-1-4503-8599-2/21/06…$15.00
https://doi.org/10.1145/3469259.3470490

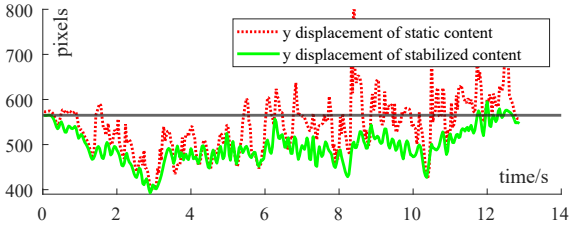

Figure 2: Y-axis (vertical) movement measured by a camera located 20 cm from our POD prototype while the latter was hovering.

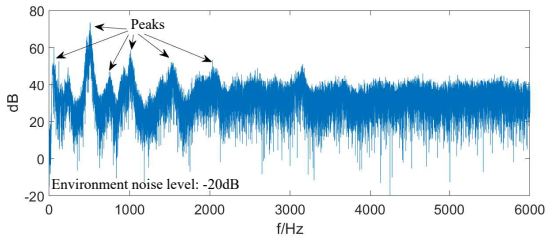

Figure 3: Noise spectrum of our POD prototype. It has multiple strong tonal components (peaks) and a relatively wide band.

smartphone apps for POD. This flexibility contrasts with other drone platforms that develop their own ecosystems, e.g., DJI and Skydio.

Hands-free "Smartphone": POD's hands-free experience opens up a host of new multitasking opportunities for mobile users in situations when holding a device in hand is either inconvenient or impossible. Users could interact with computing or engage with each other via video while cooking, walking around the house, or jogging, gardening outdoor. The latter would allow one user to have a continuous virtual presence in another user's life via POD.

Personable Interaction: There is a growing literature about human-drone interaction with a focus on how users command a drone [13], e.g., using gestures [12, 24, 26] and touch controls [9]. Few works, however, examine how a drone could communicate with users directly, without using an intermediary device like a handheld terminal. For example, Szafir, Mutlu and Fong [31] studied how a drone could use an LED ring to communicate its flight intention.

POD, with its display, provides a much richer platform for this type of direct communication. For example, POD can use smiley faces to comfort the user or indicate its intention, which is important because many users are uncomfortable with a drone flying around at such close proximity. POD can also use arrows or other symbols to direct the user's attention.

First Response and Public Safety: We also envision POD being used in the context of first response and public safety. Its display can present instructions to first responders as they navigate difficult scenarios, like fires and disasters. It could present interactive maps or other informative graphics to those trapped in hard-to-reach areas. At the scenes of car accidents, police could send POD to facilitate a remote video call with the driver and inspect their license, or to review damages and interview witnesses on the scene.

3 OPEN CHALLENGES

While drones for cinematography have been extensively studied and widely used, drones as a personal device have not. Our experience with the POD prototype has revealed technical challenges unique to personal drones, especially those related to their interacting with the user at such a close proximity.

3.1 Display Usability

Drones are dynamic systems. Keeping them stationary in the air is a long-studied and difficult problem. In order to capture high-quality videos and photos, commercial cinematographic drones have gone a long way toward solving this problem. Unlike POD, they benefit from two factors: they usually hover far away from the target scene; and the impact of instability on the footage they take can be removed by post-processing.

POD, on the other hand, must present a stationary, readable display to its user, and at a much closer distance, with the screen content often being generated in real time. Figure 2 (red, dotted line) presents how the screen of our prototype POD moves along the vertical axis during one of its flights, despite its best effort to hover, as captured by a camera. Such motion poses significant usability challenges for the display, especially for reading small text. Using a drone-carried iPad, the authors of [30] found, not surprisingly, that a moving display carried by the drone requires a larger text font for readability, although the study did not isolate the impact of the drone's inability to stay stationary in the air, as is important to POD.

3.2 Noise Suppression

Drones are noisy, and drone noise suppression has been studied extensively [22]. The noise worsens disproportionately when the user is close to the drone. And POD intends to serve its user at the intimate smartphone distance, much more closely than cinematographic drones, which acerbates the noise problem. Our POD prototype, though quite small, can produce noise up to 75 dBA, if measured from two feet away. Commercial cinematographic drones are about as noisy, or noisier, due to their larger size [3]. Noise this loud is known to be harmful of humans during long periods of exposure [8]. Cinematographic drones usually operate far from users, and noise captured by their microphones can largely be removed through post-processing. In contrast, POD operates at the smartphone distance from the user and the noise must be suppressed in real-time.

Many have explored using passive methods to reduce drone noise: covering the propellers with sound-absorbing materials and metal surfaces [23]; using ducted (or shielded) propellers [20]; using specially shaped propellers, e.g., DJI low-noise propellers [2]; and using propellers with more blades. These methods have only limited success. For example, ducted propellers [20] only reduce the noise by up to 7 dBA.

The synchrophaser system [18] is an active drone noise cancellation method. It requires that all propellers have exactly the same rotational speed and a fixed phase difference. It thus only works for fixed-wing aircraft with symmetric sets of engines. However, small rotorcraft like drones do not meet the synchrophaser requirements—their rotors constantly change speed to keep the craft stable. Noise cancellation techniques widely used in headphones can reduce noise at the ear. These techniques work well with low-frequency (< 2 KHz) and tonal noise. Unfortunately, as shown in Figure 3,

drone noise consists of tonal noise of both high and low frequencies, as well as random wide-band noise higher than 6 KHz. While active noise cancellation (ANC) has been implemented to reduce extra noise captured drone microphones [15, 34], no existing drones employ ANC to reduce the noise heard by its user, as is the goal of POD.

3.3 User Following

POD needs to present its display to a potentially mobile user. We envision that it mimics a human assistant holding the smartphone for its owner. This natural following model goes beyond the following capability enjoyed by commercial drones or reported in the literature. First, POD has to not only *follow* the user but also orient the display properly. Second, while it is tempting (and feasible) to position POD at a constant distance with a constant orientation relative to the user's head or shoulders while the user moves around, this can be annoying. Imagine a human assistant holding the smartphone for the assisted. The assistant might decide to remain stationary while the assisted turns away from the display temporarily. One could imagine there exist "eager" vs. "lazy" assistants, with a lazy assistant preferring to stay stationary and following the assisted only "reluctantly". POD must allow its user to adjust how eagerly it should behave.

3.4 Programming Interface

POD shares much of the hardware and software of smartphones and can benefit from the mature ecosystems of smartphones in app development. However, the drone's movement poses fresh programming challenges to even skilled developers. We consider two different programming support features. First, a POD application that can retrofit existing mobile apps. This retrofitting application would provide a natural interface for user input. It would not only convert speech or gestures into touch commands but also control POD's movement. For example, with this app, a POD user can use Android's built-in Camera app to take a photo after directing POD to the right position.

Another, more interesting, support feature would be a library that developers can use to command POD and create novel apps catered to POD's use cases, like those outlined in §2. Both the capabilities of the drone and the latency of the phone-drone communication constrain this library interface. For example, a programming interface that directly sets the drone's motor thrust is unsafe because the drone firmware itself stabilizes POD and must have complete control over the motors. And the (non-deterministic) delay of USB communication between the phone and the drone makes it challenging for the phone itself to implement the stabilizer. Moreover, given the physical nature of drone control, only a single app should be allowed to command the drone at a time. This suggests that the operating systems on the drone and the phone should collaborate to manage the capability of drone control as a privilege that must be explicitly acquired (and then released) by an app.

4 DESIGN AND IMPLEMENTATION

We next present our early prototype of POD, which is based on a custom-made nano-drone and a Google Pixel 4 smartphone. A video demo of our prototype interacting with a remote-controlled, life-size mannequin is available at [6].

4.1 Mechanical & Electrical

Drone: POD's drone consists of a controller board, motor components, and a 3D-printed frame. We base the controller board on the Bitcraze Crazyflie, an open-source nano-drone platform [1]. We upgraded Bitcraze's board to have more I/O ports. Unlike the Crazyflie, which uses brushed motors, POD uses 1104-4500KV brushless motors and an iFlight 12A electronic speed controller (ESC). We modify the Crazyflie firmware's motor control code to accommodate the ESC. We add a height sensor and an optical flow sensor to the drone in order to achieve autonomous hover stabilization (See §4.2). The drone is powered by its own dedicated battery and has a flight time of four minutes when carrying the phone.

Smartphone: Our prototype uses a Google Pixel 4. We remove all unnecessary parts to pare down the mass as much as possible. Only the motherboard, USB-C connector, display, and battery are kept for our experiment, totalling a mass of 63 grams. The Pixel, which can act as a USB host device, sends control packets to and receives data from the drone via one USB-C to Micro-B cable.

4.2 Autonomous Hover

Smaller drones, including the Crazyflie, do not usually come with stable autonomous hovering out of the box; a human needs to manually direct the hover via remote control. But several methods are available to enable autonomous hover. Most cinematographic drones localize themselves with GPS, which only has meter-level precision and only works outdoors. Some drones use a motion capture system consisting of multiple cameras to precisely calculate drone location within the coverage area, e.g. [4, 19]. Such motion capture systems are not only expensive but also have limited coverage area, not suitable for the use cases of POD.

Our POD prototype uses a much simpler, more mobile, solution: an optical flow sensor and phone camera. Optical flow is the pattern of apparent motion of objects due to the relative motion difference between the observer and a scene. By tracking the optical flow of the ground, a drone can know its relative horizontal displacement and make timely adjustments. We use the phone camera as a secondary observer to make the drone aware of its global environment. Unlike the optical flow sensor, the camera can help the drone stay within the user's sight.

4.3 Vision

POD uses computer vision to locate and track its user, leveraging the phone's front-facing camera and computational power, namely through PoseNet.

PoseNet [7] is based on PersonLab [25], a fully convolutional neural network implemented in TensorFlow. Because PoseNet is computationally heavy, we reduce its use with the Lucas-Kanade (LK) optical flow algorithm. The algorithm matches the human feature points of a frame with the same corner feature points from the previous frame and estimates the geometric displacement between these two sets of points. Because the LK algorithm is much more efficient than PoseNet, we invoke PoseNet to retrieve accurate feature points only every fourth frame and use the LK algorithm in between to track these points.

Human Position and Orientation Estimation: POD must estimate the user's distance and orientation in its own coordinate space (or

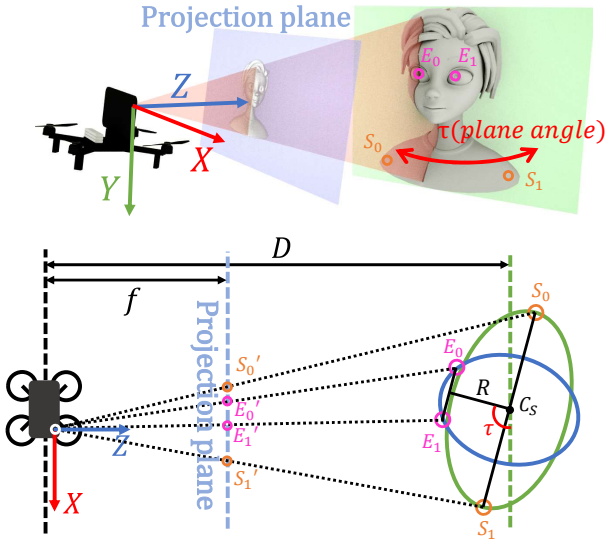

Figure 4: Spatial relationships between POD and its user: side view (top) and top view (bottom). In the side view, the red plane is part of the camera's X plane, and the green plane is the plane that passes through the shoulders, i.e., S_0 and S_1, and is perpendicular to the ground. τ represents the angle between the camera X plane (red, parallel to the plane defined by the Y and Z axes) and the user's shoulder plane (green). The projection plane is parallel to the X-Y plane and lies at a distance of f from the camera, where f is the camera's focal length.

camera coordinate space) *without* using a depth camera. Figure 4 depicts the spatial relationship between POD and its user. The distance and orientation to be estimated are marked as D and τ respectively. We assume the following parameters are known, either from calibration or being directly supplied by the user: (*i*) the focal length f of the camera; (*ii*) the distance, L_e, between the user eyes, E_0 and E_1 in Figure 4; (*iii*) the distance, L_s, between the user's shoulders, S_0 and S_1 in Figure 4; and (*iv*) the distance between the two parallel planes perpendicular to the ground defined by the shoulders and the eyes, respectively, marked R in Figure 4.

POD first measures the lengths of $S_0'E_0'$ and $S_1'E_1'$, denoted by p and q, respectively, on the projection plane, by simply counting pixels in the camera view. When the user changes their orientation, p and q change accordingly. POD derives τ from p/q using the following relationship:

$$\frac{p}{q} = \frac{\frac{(L_s - L_e)}{2}\sin\tau + R\cos\tau}{\frac{(L_s - L_e)}{2}\sin\tau - R\cos\tau}$$

POD computes D using the pinhole camera model as

$$D = f \cdot L_e \cdot sin(\tau)/L_e'$$

where L_e' is the distance between the user eyes on the projection plane, i.e., $E_0'E_1'$.

Feedback control: To accommodate many useful tasks, POD must remain at a predetermined distance and orientation (yaw) relative to the user. To maintain these requirements in real time, we employ a set of three PID controllers that are synchronized with PoseNet, running at around 50 Hz. The controllers correspond to POD's yaw, distance from the human, and position along the x-axis.

5 EARLY RESULTS

Based on our prototype described above, we next present some early results in addressing the challenges discussed in §3, with mixed success.

Evaluation setup: In order to test the prototype, we mount a human bust mannequin on a robot tank chassis driven by two DC motors, as shown in Figure 1. The mannequin has realistic features and is the actual size of a human torso—approximately 37 cm wide and 42 cm high. We use an Arduino and an ESP-01 Wi-Fi module to create a Wi-Fi remote control for the mannequin. The mannequin is recognized normally by PoseNet and contains all points of interest needed by our implementation (shoulders and eyes). Controlling the mannequin allows us to safely test our human following implementation, including much of our state machine.

5.1 Display Usability

In order to improve the usability of POD's display, we adjust the display content in real time to counter the movement of the drone. A naive implementation would use a sensor (camera or IMU) to measure the device's movement and inform the graphical rendering system to adjust content accordingly. However, the latency between sensing the movement and showing the adjusted content on the display would render this implementation inefficient and ineffective. Therefore, we experiment with several predictive stabilization methods originally intended for handheld devices, like NoShake [27]. NoShake was intended to stabilize content when a user reads a handheld display while walking. Walking causes hand and, as a result, display movement relative to the eyes. NoShake combines a spring-mass model of the display content and readings from the phone's accelerometer to move the display content opposite to the display movement so that it remains stable relative to the user's eyes.

Our experimental measurements result in Figure 2 shows that our current implementation of NoShake noticeably reduce the displacement of screen content when compared side-by-side with identical unstabilized content. Because movement of a hovering drone is different from that of a mobile user hand, i.e., less periodic and more frequent, we had to manually tune the spring factor and dampener friction constant to be effective as shown in Figure 2. We expect a data-driven, machine learning approach would be more effective. We also notice that when NoShake moves the screen content, it occasionally introduces the ghosting effect [5], partly due to the very high refreshing rate of Pixel 4's screen (90 Hz).

5.2 Noise Suppression

Because prior work [20, 23] has highlighted the limits of passive noise reduction, we focus on active methods. Active noise reduction leverages destructive interference: it generates a sound wave with the same amplitude as the target noise but with inverted phase, such that the two cancel each other out at the receiver. In POD's case, the user's ears represent the receiver.

We experiment with the feedback-based active noise cancellation setup show in Figure 5, which requires a feedback microphone to be placed next to the user's ear (two feet away from POD). We focus

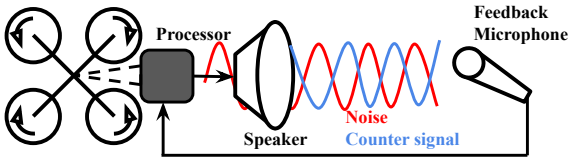

Figure 5: Our active noise reduction system consists of a speaker, a feedback microphone located near a user ear, and a processor on the drone.

User State	Recognition Method
Summoning	Raise right wrist above eye line
Relieving	Raise left wrist above eye line
Major_motion	At least one of X, Z, or orientation exceeds acceptable ranges
Minor_motion	No change in position or orientation that exceeds acceptable ranges, head begins to move while shoulders still or vice versa
Lost	No user in view

Table 1: Methods for recognizing the current human state.

on cancelling the prominent tonal components of the drone noise. First, a micro-controller (STM32F4, similar to the one used in the drone) that is connected to the feedback microphone analyzes the noise spectrum to determine the tonal noise with the highest intensity. Next, the speaker carried by the drone generates a sound wave with an adjustable phase and the same frequency as the high-intensity noise. The microphone provides feedback to the speaker, which can then adjust the wave's phase to perfectly cancel the noise.

A decibel meter next to the feedback microphone shows that the above active cancellation technique reduces the noise from 73 dBA to 70 dBA. While small, this reduction is promising since we only used a single speaker and only cancelled the strongest tonal component. We plan to investigate the use of multiple speakers. Most noise cancellation solutions [38] use multiple speakers and cancel more frequency components.

5.3 User Following

We experiment with configuring POD to mimic a human assistant holding a smartphone for the assisted. Figure 6 represents POD's behavioral model as a state machine. POD changes its state based on its understanding of the user's state. Table 1 summarizes what user states POD recognizes and how. The behavioral model incorporates a tunable parameter, T, which can range from a fraction of a second to tens of seconds, depending on the application. A larger T leads to a "lazier" POD that follows its user more reluctantly.

In `Home`, POD actively maintains a predetermined distance and orientation with respect to the user's shoulders, using the feedback controller described in §4.3 to follow the user when they make major movements as outlined in Table 1. POD ignores any minor movements. In `Idle`, POD does not move at all. POD enters `Idle` after the user rotates past the predetermined τ threshold. It returns to `Home` after T elapses.

The user can use the relieving and summoning gestures to indicate that it would like POD to immediately enter `Await` or re-enter `Home`, respectively. In `Await`, POD only yaws to keep the user in its camera view. That is, it only activates the feedback controller for yaw.

A short video showing our prototype eagerly following our test mannequin is available here [6].

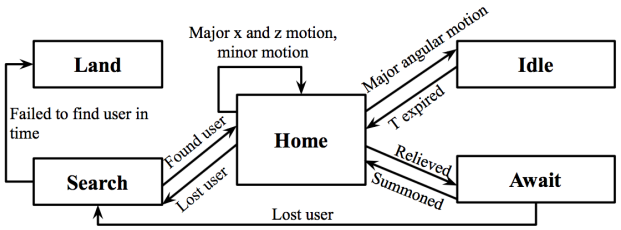

Figure 6: The behavioral model of POD. State transition is based on the perceived state of the user. In `Home`, POD actively follows the user and presents its display at a pre-determined distance and angle relative to their shoulders. In `Idle`, POD holds still. In `Await`, POD yaws to keep the user in the frame.

5.4 Programming interface

Our prototype's phone uses the Android USB library to control the drone via a connecting cable. The drone used in our prototype can only determine its height (z-axis) accurately. It can estimate changes in the x-y plane with the optical flow sensor with less accuracy.

We currently support a programming interface allowing an Android app to directly command POD's absolute and relative movement along the z-axis (height) and its relative movement in the x-y plane. These functions are complete in that an POD app developer can use them to command POD to any position relative to its user, especially with feedback from POD's camera-based human position and orientation estimation (§4.3). Implementing this USB programming interface requires a new periodic task in the drone firmware. The task runs at 100Hz, which is one-tenth of the main controller loop frequency. We also plan to expand our interface by converting our human position and orientation estimation technique (§4.3) into an Android library.

The programming interface is blocking, or synchronous. That is, once the phone app invokes an API, it will block until the drone completes the requested movement and returns a message. We choose this synchronous design because we believe the app should move the drone one command a time, sequentially, for safety. Asynchronous motion APIs could lead to unpredictable and dangerous behavior.

6 RELATED WORKS

Drone as User Interface: Many have explored harnessing drones to design new mobile user interfaces like flying 3D displays [28], haptic interfaces [36], and projectors [11, 16, 29, 33, 39]. DisplayDrone [28] carries a flexible touch display with remote tele-presence functionality. ISphere [35] is a flying, omni-directional, spherical LED display. The authors of [30] studied text readability on a tablet computer carried by a drone. These works were partially geared towards presenting information and guidance to mobile users. Their focus, however, was not on system design and implementation. Their prototypes offer no communication between the display content and the drone. In contrast, POD tightly integrates the drone and the phone and depends heavily on collaboration between them.

Human Following by Robots: There is a growing literature that studies how mobile robots can track and follow human users. While POD builds on many ideas presented by past works, it uniquely features a rich user interface with both input and output flow. It thus

faces a new set of challenges. Existing robotic followers, including drones [10, 21], usually aim at staying within a certain distance of their human users. POD must additionally orient the display properly and adjust screen content in real time as described in §5.1. Drone.io [14] employs a user-following drone to project a user interface on a large surface as output and allows gesture-based interaction. The use of projection, instead of a display, leads to several important key differences from POD. First, users interact with POD directly, while they interact with a projected screen in Drone.io. Second, the drone in Drone.io operates at a much larger distance from the user. As a result, Drone.io does not face the same challenges in user following, noise or display stabilization as POD. More importantly, the focus of [14] is on the design and evaluating Drone.io as a novel interface for human-drone interaction. The reported prototype is based on commercially available drones that can be controlled via a limited interface. Indeed, the projector and a computing element (an iPod touch) carried by the drone do not communicate with the drone in the reported prototype. In contrast, we focus on the design and implementation of POD itself and custom-built a drone for tight integration with the carried smartphone.

The Georgia Tech Miniature Autonomous Blimp (GT-MAB) [37] can follow and track a human user using a camera and multiple thrusters. While GT-MAB and POD accept similar user input and both use vision-based sensing, GT-MAB has no output or display. Unlike drones, blimps do not rely on motorized propellers for lift, so they are quieter, more stable but much harder to maneuver. The primary challenge faced by GT-MAB (or a blimp-based POD) is its size: to carry the same reduced smartphone system (of 63 gram) as used in our prototype, GT-MAB must gain a volume of 0.6 m^3.

ACKNOWLEDGMENTS

This work was supported in part by Yale University, NSF Award 2016422 and its REU supplement. We are grateful to the anonymous reviewers for their input, especially the Tinker Bell example. Grace Cheung helped with PoseNet on Pixel 4; Nina Bernick helped with the camera-based evaluation setup and display stabilization.

REFERENCES

[1] Bitcraze - crazyflie. https://github.com/bitcraze/crazyflie-firmware.
[2] DJI low noise propellers. https://store.dji.com/product/mavic-2-low-noise-propellers.
[3] Drone noise level. https://www.airbornedrones.co/drone-noise-levels/.
[4] Flying machine arena. https://www.flyingmachinearena.ethz.ch/.
[5] Ghosting (television). https://en.wikipedia.org/wiki/Ghosting_(television).
[6] Our prototype's human following test. https://youtu.be/_ZVm9seYu1o.
[7] Posenet. https://github.com/tensorflow/tfjs-models/tree/master/posenet.
[8] What noises cause hearing loss? https://www.cdc.gov/nceh/hearing_loss/what_noises_cause_hearing_loss.html.
[9] Abtahi, P., Zhao, D. Y., E, J. L., and Landay, J. A. Drone near me: Exploring touch-based human-drone interaction. In *Proc. Int. Conf. Ubiquitous Computing (UbiComp)* (2017).
[10] Boschi, A., Salvetti, F., Mazzia, V., and Chiaberge, M. A cost-effective person-following system for assistive unmanned vehicles with deep learning at the edge. *Machines* (2020).
[11] Brock, A. M., Chatain, J., Park, M., Fang, T., Hachet, M., Landay, J. A., and Cauchard, J. R. Flymap: Interacting with maps projected from a drone. In *Proc. ACM Int. Symp. Pervasive Displays (Perdis)* (2018).
[12] Cauchard, J. R., E, J. L., Zhai, K. Y., and Landay, J. A. Drone & me: an exploration into natural human-drone interaction. In *Proc. Int. Conf. Ubiquitous Computing (UbiComp)* (2015), pp. 361–365.
[13] Cauchard, J. R., Khamis, M., Garcia, J., Kljun, M., and Brock, A. M. Toward a roadmap for human-drone interaction. *ACM Interactions* (2021).
[14] Cauchard, J. R., Tamkin, A., Wang, C. Y., Vink, L., Park, M., Fang, T., and Landay, J. A. Drone.io: A gestural and visual interface for human-drone interaction. In *Proc. ACM/IEEE Int. Conf. Human-Robot Interaction (HRI)* (2019), pp. 153–162.
[15] Chun, C., Jeon, K. M., Kim, T., and Choi, W. Drone noise reduction using deep convolutional autoencoder for uav acoustic sensor networks. In *Proc. IEEE MASSW* (2019).
[16] Darbar, R., Roo, J. S., Lainé, T., and Hachet, M. Dronesar: Extending physical spaces in spatial augmented reality using projection on a drone. In *Proc. ACM Int. Conf. Mobile and Ubiquitous Multimedia (MUM)* (2019).
[17] Funk, M. Human-drone interaction: let's get ready for flying user interfaces! *ACM Interactions 25*, 3 (2018), 78–81.
[18] Jones, J. D., and Fullert, C. R. Noise control characteristics of synchrophasing part 2: Experimental investigation. *AIAA Journal* (1986).
[19] Kushleyev, A., Mellinger, D., Powers, C., and Kumar, V. Towards a swarm of agile micro quadrotors. *Autonomous Robots 35*, 4 (2013), 287–300.
[20] Malgoezar, A., Vieira, A., Snellen, M., Simons, D., and Veldhuis, L. Experimental characterization of noise radiation from a ducted propeller of an unmanned aerial vehicle. *International Journal of Aeroacoustics* (2019).
[21] Mao, W., Zhang, Z., Qiu, L., He, J., Cui, Y., and Yun, S. Indoor follow me drone. In *Proc. ACM Int. Conf. Mobile Systems, Applications, & Services (MobiSys)* (2017), pp. 345–358.
[22] Miljković, D. Methods for attenuation of unmanned aerial vehicle noise. In *Proc. MIPRO* (2018).
[23] Mohamud, A., and Ashok, A. Drone noise reduction through audio waveguiding. In *Proc. ACM Wrkshp. on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet)* (2018).
[24] Obaid, M., Kistler, F., Kasparavičiūtė, G., Yantaç, A. E., and Fjeld, M. How would you gesture navigate a drone? a user-centered approach to control a drone. In *Proc. Int. Academic Mindtrek Conference* (2016).
[25] Papandreou, G., Zhu, T., Chen, L.-C., Gidaris, S., Tompson, J., and Murphy, K. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proc. European Conference on Computer Vision (ECCV)* (2018).
[26] Peshkova, E., Hitz, M., and Kaufmann, B. Natural interaction techniques for an unmanned aerial vehicle system. *IEEE Pervasive Computing 16*, 1 (2017), 34–42.
[27] Rahmati, A., Shepard, C., and Zhong, L. NoShake: Content stabilization for shaking screens. In *Proc. IEEE PerCom* (2009).
[28] Rubens, C., Braley, S., Gomes, A., Goc, D., Zhang, X., Carrascal, J., and Vertegaal, R. BitDrones: Towards levitating programmable matter using interactive 3d quadcopter displays. In *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI)* (2016).
[29] Scheible, J., Hoth, A., Saal, J., and Su, H. Displaydrone: A flying robot based interactive display. In *Proc. ACM Int. Symp. Pervasive Displays (Perdis)* (2013).
[30] Schneegass, S., Alt, F., Scheible, J., and Schmidt, A. Midair displays: Concept and first experiences with free-floating pervasive displays. In *Proc. ACM Int. Symp. Pervasive Displays (Perdis)* (2014).
[31] Szafir, D., Mutlu, B., and Fong, T. Communicating directionality in flying robots. In *Proc. ACM/IEEE Int. Conf. Human-Robot Interaction (HRI)* (2015).
[32] Tezza, D., and Andujar, M. The state-of-the-art of human–drone interaction: A survey. *IEEE Access 7* (2019), 167438–167454.
[33] Toyohara, S., Miyafuji, S., and Koike, H. [poster] arial texture: Dynamic projection mapping on drone propellers. In *Proc. IEEE ISMAR-Adjunct* (2017).
[34] Wang, L., and Cavallaro, A. A blind source separation framework for ego-noise reduction on multi-rotor drones. *IEEE/ACM Transactions on Audio Speech and Language Processing 28* (2020), 2523–2537.
[35] Yamada, W., Yamada, K., Manabe, H., and Ikeda, D. ISphere: Self-luminous spherical drone display. In *Proc. ACM Symp. User Interface Software & Technology (UIST)* (2017).
[36] Yamaguchi, K., Kato, G., Kuroda, Y., Kiyokawa, K., and Takemura, H. A non-grounded and encountered-type haptic display using a drone. In *Proc. ACM Symp. Spatial User Interaction (SUI)* (2016), p. 43–46.
[37] Yao, N., Anaya, E., Tao, Q., Cho, S., Zheng, H., and Zhang, F. Monocular vision-based human following on miniature robotic blimp. In *Proc. IEEE ICRA* (2017).
[38] Zhang, W., Hofmann, C., Buerger, M., Abhayapala, T. D., and Kellermann, W. Spatial noise-field control with online secondary path modeling: A wave-domain approach. *IEEE/ACM Trans. Audio, Speech and Lang. Proc. 26*, 12 (Dec. 2018), 2355–2370.
[39] Zhang, X., Braley, S., Rubens, C., Merritt, T., and Vertegaal, R. Lightbee: A self-levitating light field display for hologrammatic telepresence. In *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI)* (2019).