Shaping and Being Shaped by Drones: Programming in Perception–Action Loops

Mousa Sondoqah
mousa.sondoqah@mail.polimi.it
Politecnico di Milano
Milan, Italy

Fehmi Ben Abdesslem
fehmi.ben.abdesslem@ri.se
Department of Computer Science,
RISE Research Institutes of Sweden
Stockholm, Sweden

Kristina Popova
kpopova@kth.se
Media Technology and Interaction
Design, KTH Royal Institute of
Technology
Stockholm, Sweden

Moira Mcgregor
mcgregor@dsv.su.se
Department of Computer and Systems
Sciences, Stockholm University
Stockholm, Sweden

Joseph La Delfa
josephld@kth.se
Robotics, Perception and Learning,
KTH Royal Institute of Technology
Stockholm, Sweden

Rachael Garrett
rachaelg@kth.se
Media Technology and Interaction
Design, KTH Royal Institute of
Technology
Stockholm, Sweden

Airi Lampinen
airi@dsv.su.se
Department of Computer and Systems
Sciences, Stockholm University
Stockholm, Sweden

Luca Mottola
luca.mottola@ri.se
Politecnico di Milano and RISE
Research Institutes of Sweden
Milan, Italy

Kristina Höök
khook@kth.se
Media Technology and Interaction
Design, KTH Royal Institute of
Technology
Stockholm, Sweden

ABSTRACT

In a long-term commitment to designing for the aesthetics of human–drone interactions, we have been troubled by the lack of tools for shaping and interactively *feeling* drone behaviours. By observing participants in a three-day drone challenge, we isolated components of drones that, if made transparent, could have helped participants better explore their aesthetic potential. Through a bricolage approach to analysing interviews, field notes, video recordings, and inspection of each team's code, we describe how teams *1)* shifted their efforts from aiming for seamless human–drone interaction, to seeing drones as fragile, wilful, and prone to crashes; *2)* engaged with intimate, bodily interactions to more precisely probe, understand and define their drone's capabilities; *3)* adopted different workaround strategies, emphasising either training the drone or the pilot. We contribute an empirical account of constraints in shaping the potential aesthetics of drone behaviour, and discuss how programming environments could better support somaesthetic perception–action loops for design and programming purposes.

CCS CONCEPTS

• **Human-centered computing → Systems and tools for interaction design**.

KEYWORDS

drones, programming tools, soma design

ACM Reference Format:
Mousa Sondoqah, Fehmi Ben Abdesslem, Kristina Popova, Moira Mcgregor, Joseph La Delfa, Rachael Garrett, Airi Lampinen, Luca Mottola, and Kristina Höök. 2024. Shaping and Being Shaped by Drones: Programming in Perception–Action Loops. In *Designing Interactive Systems Conference (DIS '24), July 1–5, 2024, IT University of Copenhagen, Denmark*. ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3643834.3661636

DIS '24, July 1–5, 2024, IT University of Copenhagen, Denmark

ACM ISBN 979-8-4007-0583-0/24/07.
https://doi.org/10.1145/3643834.3661636

1 INTRODUCTION

In a long-term commitment to designing for the aesthetics of human–drone interactions, designing drones for the opera stage, [15, 16], drones for dancing [41], and drones that can modify their behaviour to become uniquely entangled with their user [39, 42, 43], we have approached this theme as a multidisciplinary team combining competence in interaction design, human–computer interaction, embedded systems, and mobile robotics. The complexities of programming drones, shaping and *feeling* the potential aesthetics of their behaviour, as well as handling their hardware and design, has posed interesting challenges to our design team over the years. We have increasingly identified a need for bodily ways of understanding and 'programming' drone behaviours rather than having to take a detour via software programming, signal processing, and embedded programming changes, done on a separate computer. When it is made possible to move and be moved by the drones in a direct, perceptual sense, it may become easier to *feel* the potential aesthetics that could be shaped.

In order to document how *pilots* come to understand drones and to shape human–drone interaction (HDI) by 'programming

through the body', we decided to expand our approach beyond solely observing our own design processes. We arranged a three-day drone challenge where participants with little to no prior experience of programming and piloting drones, were given limited time to explore and design interactions with an aerial nano-drone. To focus the design space further, we asked participants to work toward the specific objective of piloting their drone through an obstacle course quickly and precisely. While an intensive three-day design journey does not attain the depth of our previous projects – with years, rather than days, spent on each drone design – we argue that some of the core issues still emerged. What is more, the constrained format of the challenge made the issues more readily observable, allowing us to report on them here.

The aerial nano-drones we used are a fascinating and evocative type of robot. They have a material body [13] that moves and makes noise, tempting us to interact with them as if they were more communicative than they are. Prior accounts of design processes focused on human—drone interaction have featured encounters between designers, with their designerly aims and fleshy bodies, and drones, with unpredictable technological affordances and fragile plastic bodies [22, 50], illustrating that as these encounters unfold, designers shape not only the technology but also themselves, altering how they move to fit with the drone behaviour [16, 40].

In contrast to prior work that concerns human—drone observation [34, 52, 64] or human—drone gestures [1] [30, 37, 67], we were especially keen to observe the coupling (or breakdowns) in perception and action when controlling a drone with bodily inputs [23, 32, 62]. Our focuses on *the aesthetics of the mutual shaping of pilots and drones*, that is, how participants in the challenge were shaping drone behaviour and, in turn, became shaped in bodily ways by the drones' limitations. Our aim was to get at the bodily, felt experiences of engaging with the drones, including the moments when there were breakdowns because of the lack of access to the inner workings of the drones.

Our bricolage analysis draws upon interviews, field notes featuring both observations and organisers' own experiences, video recordings, and inspection of the teams' code. Three key observations help deepen our understanding of what it took for our pilots to get closer to the drones and to shape their behaviours:

(1) We depict how participants' understanding of drones transitioned from inspirational imagery of seamless human–drone interaction to lived experience of drones as fragile, apparently wilful, and prone to crashes. In the process of this transformation of their perception of drone technology, our participants encountered the *visibility problem* [44], common in embedded programming: as the internal system states are not visible to the external user, it is hard to disentangle or explain the different reasons why the drone may be behaving in an erroneous or unexpected way.
(2) We describe how participants came to know their drones better through intimate, bodily interactions which allowed them to more precisely probe, understand, and determine the capabilities of their drone. The iterative process of getting to know the drone unfolded through collaboration among team members – and also with the support of the organising team who provided technical support.
(3) Through discussing teams' differing approaches to building up their drone interactions – shaping and being shaped by these interactions – we illustrate the difference between *training the drone*, for instance by applying *defensive coding* in the form of programs that operate cautiously in anticipation of possible issues, and *training the pilot*, that is, changing and rehearsing the pilot's own movements to fit with existing drone capabilities.

We contribute an empirical account of shaping and being shaped by drones and lay out opportunities to better support the design of human–drone interactions. We will be drawing upon insights from the drone challenge, and the subsequent redesign we did of the drones for a second challenge organised a year later, but also further drone design projects our team members have worked on. We conclude by discussing, first, the need for easy-to-approach programming tools that allow pilots to adequately change the code or modify sensor filters of the drones in a more direct and transparent manner. Second, we note that drones may simultaneously 'train' their pilots by providing somatic signs, signals, and feedback that may be probed and felt in real-time [41]. These enhancements would support pilots in learning how they might shape their own movements to better partner with the drone, until they develop a new way of moving together with the drone – exploring and exploiting their aesthetic potential. As we discuss in another publication [59], our work with the drone challenge – and the lessons we have learned through our multidisciplinary collaboration – have implications also on the design and development of embedded programming systems and their corresponding run-time support.

2 BACKGROUND

Let us, first, provide a brief overview of drone piloting from a technical standpoint before discussing prior research on the potential somaesthetic experiences of human–drone interaction (HDI).

2.1 Drone Piloting

Conventionally, drone piloting may be achieved in three ways:

First, a control station may wirelessly connect to the drone to issue high-level piloting commands, such as "move forward 1 m", or "rotate 90° right". The control station runs a set of programs that implements arbitrary application logic, for example, to achieve a given coverage of a geographical area, achieving *fully autonomous* behaviors [20]. Most professional drone platforms operate this way.

Alternatively, a piloting device can be *manually* operated by a trained pilot, who has full control of the drone. This form of piloting is challenging. The set of knobs and handles available on a piloting device is both small and large. It is small in that modern drones offer a multitude of operating modes, and each such mode may require a separate set of knobs and handles. It is, at the same time, large for a human to control in real-time while maintaining eye contact on the drone, as dictated by current regulations.

Halfway between the two extremes lie the many solutions that offer some form of *assisted* drone piloting. Some form of piloting device is used, here, as well, like a mobile app on smartphones. The

[1] Here, we use the word *gesture* as a classified set of movements interpreted as a command, later in the paper we use the word in a different sense to describe the actions of the team members.

piloting device offers ways to achieve some degree of manual control, while the drone autonomously performs other piloting actions in support, such as keeping the position hovering, if no piloting input is received. Most consumer and pro-sumer drone platforms adopt this approach. This is the type of drone our challenge focused on (for more details, see Sec. 3.2).

The inputs received from either the control station or a piloting device are processed by a piece of embedded software, running aboard the drone, called *flight controller* [7]. Its job is to realise the movements requested by the control station or by the pilot, translating their inputs into operational settings for the motors.

2.2 Somaesthetic Human–Drone Interaction

Tezza and Andujar [61] describe how drones expanded from military operations to a range of civilian applications. The social-cultural implications [46] of this expansion prompted a surge of research into HDI [10, 21], particularly in the design space of social drones [5]. Much research into the relational aspects [29] of drone technology in these settings has focused on so-called 'natural' HDI [8], integrating gestures and movements [9], and studying emotions in relation to the design of such systems [11, 28, 64]. For example, drawing upon Communication Studies to advance HRI, Urakami and Seaborn [63] have suggested a series of nonverbal codes that address the five human sensory systems to promote more natural, inviting, and accessible experiences.

Joining scholars who challenged straightforward notions of naturalness [14, 33, 49, 60], we, instead, approach HDI with a somaesthetic sensibility. Starting from our somas – the lived and felt body as it exists, moves, and senses in the world [57] – we explore what novel aesthetic experiences can be spurred by the design of human–drone interaction. We see a somaesthetic perspective as generative for both designing and analysing human—drone interaction. A soma design stance invites us to address and change the habitual and limiting ways in which we move [33, 56]. When we interact closely with drones, we have to adapt ourselves in how we control them and move around them. This happens when using them for work purposes [36, 58], as part of leisure activities [4, 35], artistic performances [16, 37, 38], or in family settings [22, 24]. Here, we draw on our own prior work on HDI in artistic performance [15, 16], Tai Chi-inspired movement [40, 41], design processes [23, 50], and somaesthetic human–machine interaction [42, 43].

These prior works offer scaffolding for studying movement-based interactions with drones, as they consider both the somatic aspects of human–drone interaction as well as the social settings where such interaction takes place. For example, Popova et al. [50] explored possibilities for designing relationships with *"drones as 'the other' – a distinctive and separate entity, which is neither completely controlled, nor fully autonomous"*. Through interactional work within the assemblage of humans and drone technology, Popova et al. [50] opened up a design space in which to engage with an unfamiliar technology, non-habitual design activities, and exploratory ideas. Their account of an exploratory design process that unfolded over a significant period of time provides an interesting contrast to the drone challenge we focus on in this paper, in that the participants in our challenge were brought to explore and develop human–drone interaction within the limited time frame of a three-day event, without dedicated training in interaction design or somaesthetics, and toward the specific objective of piloting the drone through a competitive obstacle course.

3 DRONE CHALLENGE

The drone challenge was set up as a three-day hackathon, yet without the overly stressful, competitive elements typical of hackathons. It took place in a unique space, an abandoned reactor hall deep in the bedrock underneath the Royal Institute of Technology (KTH) campus in Stockholm, Sweden.

3.1 Objective and Rules

The challenge was to fly a *nanodrone* through an obstacle course set in a *drone arena*, exemplified in Fig. 1, in a fixed time while collecting as many markers as possible. Drone piloting was achieved by a competitor-participant who physically interacted with the drone in the arena, for example, by making gestures that were detected by onboard *proximity sensors*. Collecting a marker was achieved by successfully flying the drone over a marker at any height.

The teams implemented the drone control logic running on a control station as a Python program. The program determined how a drone would react to a human pilot's gestures. For example, the program steers the drone sideways when a participant moves their hand closer to the drone on one side. This is a form of *assisted* drone piloting, achieved by interacting with the drone physically. Only one human pilot at a time could be active in the arena during a run. During runs, participants were not allowed touch the drone or install anything in the arena, such as additional markers or sensors.

We purposefully set up the event so as to discourage participants from taking an unnecessarily competitive or rushed approach. In introducing the event to the participants, we stated learning about the drones and having fun as the main goals of the challenge rather than winning, although we did explain that there would be prizes for the most successful teams. Framing the challenge around learning and emphasising that the participants were free to decide how much time they wanted to spend working on the challenge, we tried to establish a calm and supportive atmosphere. We did this to center values like curiosity and collaboration, rather than aggressive rivalry and competitiveness. The teams had to leave the premises within reasonable time each day to be transported up to the ground from the reactor hall 12 meters down. No one worked through the night or was asked to complete tasks in a stressful manner.

3.2 Technology

We used the Crazyflie 2.1 (see Fig. 2), an open source nano-drone platform that only weighs 27 g and fits in the palm of a hand. The Crazyflie offers a complete development environment including a piloting GUI, a rich set of application programming interfaces, detailed documentation, and tutorials. We also provided examples of the control logic running on the control station that showed participants how to achieve simple piloting functionality, such us "pushing" the drone in a given direction when the drone's onboard sensor detects that someone's hands are approaching, or "pulling" the drone when the hands are withdrawing.

An essential element to achieve this functionality is localising the drone in a 3D reference system. As the challenge took place in

Figure 1: The drone arena, located in the former reactor hall under the campus of KTH in Stockholm. One of the two identical obstacle courses lies in the foreground, with markers and obstacles of various shapes and sizes.

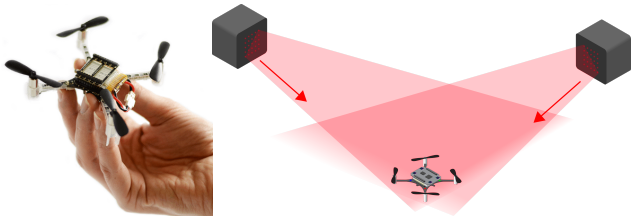

Figure 2: Left: Crazyflie nano-drone. Right: The Lighthouse positioning system - infrared lasers that monitor the flight environment.

an indoor setting 12 meters underground, competitors were unable to rely on GPS. Instead, we equipped the drone arena with the Lighthouse localization system. This system used two base stations deployed at opposite corners of the arena to emit infrared laser scans (see Fig. 2). These were detected by the drone using dedicated sensors. Based on the difference in time of arrival between the scans from different base stations, the drone technology could estimate its position. The Lighthouse localization system was sufficiently simple to install in a temporary location and provided a user experience largely similar to other indoor localization systems, including optical ones, such as OptiTrack [1].

The Lighthouse localization system worked reliably only as long as the laser scans were able to constantly reach the drone, similar to an OptiTrack system requiring line of sight to the markers aboard the drone. If the path from a base station to the drone got occluded, say because a person moved in between the two, the drone temporarily lost the base station inputs. Depending on the number of base station occlusions and the duration of the disruption, the drone either became temporarily unstable (yet eventually reclaimed a stable behavior), or crashed due to having lost positioning information completely. The operation of the localization system was a significant emergent factor in shaping relationships between pilots and drones during the challenge.

3.3 Schedule and Teams

The event began with a kick-off at lunch time, including an *opening talk* to inspire the participants, as well as a *a two-hour tutorial* on programming the Crazyflie, using teaching material developed specifically for the event. By the end of the tutorial, even teams with no previous experience of the technology were able to fly their assigned drone. The remainder of the first day and the entire second day were, then, devoted to *challenge trials*. Teams were free to work shorter or longer hours as they preferred, to program their drone to interact with their nominated pilot in the arena.

We set up two separate drone arenas for the trials. Experts on our team were available to provide technical support throughout the trials. They spent much of their time helping the teams resolve technical issues before they submitted their code for the final challenge. The third day of the event began with the challenge, observing the objectives and rules outlined in Sec. 3.1. Each team was allowed four attempts to complete the obstacle course. The event concluded with with a short prize ceremony and lunch.

Participation was open to anyone free-of-charge irrespective of age, profession, and past experience with drones, on a first-come-first-served basis. When circulating the call for participation, we advertised prizes for the most successful teams, including wireless headphones, movie tickets, and the opportunity to keep the drone the team had worked with. As we provided all necessary drone hardware, a team only needed to bring a computer for programming. While the challenge was advertised as open to all, the details of the challenge likely made it more compelling to those with some familiarity with drones and/or at least some programming skills. The timing of the challenge during three weekdays in June, at the end of the semester, along with its location on a university campus, also likely played a role in making it particularly attractive to local university students, especially from engineering programs.

Of the six teams that signed up, five participated fully, resulting in a total of 22 participants. All teams but one were entirely composed of university students. Most participants had previous programming experience in a variety of languages, including Python, but only a few had any prior experience of drones, e.g. in activities like aerial filmography that had little overlap with the challenge.

4 MATERIAL AND METHODS

Next to arranging the challenge as a demonstration of movement-based drone piloting and an opportunity for participants to learn about drones, we approached it as a means to study HDI.

Our **data collection** encompassed multiple methods. First, we took *field notes*, detailing our observations and experiences. These include notes from both the researchers who served as technical support – and hence heard first-hand about issues the teams were facing and the solutions they were resorting to – and the rest of the team whose main task was to observe and record the event, drawing upon their training in the social sciences and human–computer interaction. Second, the teams needed to submit *the code* they produced for controlling the drone ahead of the challenge. This gives us a detailed view into the design choices they made and provides evidence of some of the technical options that were explored but ultimately abandoned. Third, we *video-recorded* large parts of the challenge. We used stationary cameras to capture the activity in each of the arenas during the trials and the challenge. We used smartphones to capture shorter instances of teams' activities and interactions by their desks and in the event space. Fourth, two researchers conducted short *group interviews* with the teams, asking about their backgrounds, their motivations for participating, and their experiences of the challenge. These complement the insights into the teams' experiences we gained through observation.

We conducted our **data analysis** in the spirit of *bricolage* [54]. The code and the field notes, taken together, provide a rich description of the challenge. We worked collaboratively to reflect on observations from revisiting field notes and inspecting the code [17], scrutinising overlaps and distinctions in what caught different researchers' attention during initial data analysis. Through this, we identified *the mutual shaping of pilots and drones*, that is, how participants adapted the drones to match their aims but also how the participants adapted themselves so as to interact with the drones, as a core theme for closer analytic development in which we included also the interview and video data.

We attended to **ethical research practice** by following standard procedures for informed consent and through conversation among the authors[2]. All participants were given an information sheet as they first arrived to the event. After having had the time to read it and ask any questions, they gave written consent for their participation in the study, with the understanding that the authors would (1) observe the challenge, (2) video record parts of it, and (3) invite participants to take part in interviews, which they could freely choose to join or decline, without any repercussions. We also explained to participants that they could withdraw their participation at any point, without needing to provide a reason, and that we would remove their data from our study if requested. We use pseudonyms when referring to participants and teams.

5 FINDINGS

Our findings are three-pronged. First, we report on how participants' understanding of drones transformed from the early inspirational imagery of seamless human–drone interactions to the reality of fragile, wilful drones, prone to crash. Second, we illustrate participants' bodily interactions to make sense of drone capabilities. Third, we depict teams' explorations in building up their drone interactions, considering the spectrum between shaping drone behaviours to give the pilot more leeway versus changing pilots' behaviours to fit with the drone.

5.1 Learning about Drones

We, now, illustrate the multiple narratives around drones that emerged during the challenge. We pay particular attention to how these narratives were reflected in participants' programs, and how they influenced teams' development of interactions with their drone.

First, the message conveyed during the opening talk was uplifting, evocative, with a focus on success stories. The speaker emphasised the need to establish a *shared body language* between humans and drones. The slides shown to the participants were void of any technical detail. The speaker drew attention to the challenges in establishing synchrony between people and robots, acknowledging the difficulty of building a shared language: *"When we see people in the movies moving with a robot, with robot bodies that look like our own, it is very easy to imagine what this is about. But what do you do when the robot looks different to you? How do you control, for example, sensing and actuating for additional limbs that have hundreds of joints? The movies make it look really easy but there is a lot that has to happen between your familiar body and this weird robot-body."* However, the speaker did not get into the technical details of how to achieve such interactions. Instead, he demonstrated the development of shared language through videos showing gestures implicitly commanding drones, akin to what the participants needed to accomplish in the challenge. Not a single crash or malfunction was shown at any point during this presentation.

Second, in striking contrast to the opening talk, the subsequent tutorial focused almost exclusively on *technical details* (see Fig. 3). It was centered around possible issues and how to avoid them. Even the simplest operation, such as getting the drone to take off, was narrated as an activity that requires careful preparations. These

[2]The study does not fall within the purview of ethical review in the country where the authors work.

Figure 3: The programming tutorial that outlines how each teams can edit the code and replace broken propellers on their drones.

Figure 4: Left: a team of pilots examines code on a laptop. Right: discussions during the trials in the drone arena.

include both mechanical aspects, such as assembling the drone in the right way, balancing it manually, and checking it meticulously before turning it on, and actions related to software, since the development environment must be set up correctly and connected wirelessly to the drone, before issuing any command. Roughly half of the time of the tutorial went into describing what could go wrong and how to work around those issues. The speakers conveyed lots of "hands-on" experience of the sort that is rarely available in manuals.

Third, discussions during the trials (see Fig. 4), took place between the participating teams and our technical support. They were mostly related to questions about how to understand drone behaviors in the many cases when the drone did not behave as expected, the reasons for why the drone crashed, and how to repair it when something broke. Participants came to the technical experts with many questions:

> *How come the drone was hovering stably for several seconds and suddenly lost control?*
>
> *Even without imparting any command, the drone keeps moving around... why?*
>
> *How can I see at all what the readings of the onboard sensors are, to check what the drone is doing?*

The technical experts' responses acknowledged the high degree of unpredictability common for drones:

> *"There may be different reasons why it doesn't take off. Sometimes the best solution is to turn it on and off."*
>
> *"It is really super random. Sometimes it could work, sometimes it doesn't."*

These situations are arguably an instance of the *visibility problem* [44], common in embedded system programming. Resource-constrained embedded systems, such as drones, often do not run full-fledged operating systems and, hence, lack most of the inspection and debugging facilities. The nano-drone, Crazyflie, adopts a form of assisted flight (see Sec. 2), and the flight controller aboard the drone runs on bare hardware. Understanding the execution of programs on a device that has no operating system and only a few LEDs for debugging is often compared to *"looking at an elephant through a keyhole"* [44]. Tools exist to address this problem [65] but they are hard to approach for beginners.

During the challenge, the issues experienced by our participants arose mainly due to two factors. First, the sensors aboard the drone used to detect human gestures are both slow and imprecise. The time it takes to detect a nearby obstacle may reach up to a few seconds, defeating the illusion that a mutual relationship between the drone and the pilot could develop with the same time dynamics as between people. Second, the localization system (see Sec. 3.2) requires constant line of sight between the base stations and the drone, or the latter loses control and crashes. The participants did not immediately realise the consequences of this and how the pilot needed to this feature into account.

As the technical experts progressively answered the participants' questions and explained the reasons why problematic situations occurred, participants entered a mutual adaptation loop, adapting the programs piloting the drones and/or their own body movements, depending on what factor they considered easier to handle.

The transition from the early representation of smooth and expressive human–drone interaction in the opening talk, to brittle behaviors and crashes in the arena was abrupt. This shift in perspective is apparent also in the participants' programs: Through applying established code inspection techniques [17], we were able to shed further light on how the different narratives around drones manifest in participants' coding. As concrete evidence, Appendix A reports an excerpt of the program by the team *Bellman Brothers*.

Many teams began with high ambitions, aspiring to the forms of HDI shown during the opening talk. Those interactions were rich and smooth, yet what was not apparent in the opening talk were a number of technical features that made the input to the flying drone much more accurate than the on-board navigation sensors that we asked our participants to use. Participants came to realise this discrepancy through trial and error.

After multiple crashes and conversations with the technical team, participants turned to dealing with the inaccuracy of the onboard sensors. This is where they started commenting out portions of code that were originally meant to replicate the human–drone dynamics seen in the opening talk. This is evidenced, for example, in the existence of multiple versions of the function **get_speed_command** in *Bellman Brothers'* program (Appendix A), or in their work on the **which_region** function, which is defined and implemented, but ultimately *never used*. These functions were eventually considered not suited to deal with the unreliable inputs and, thus, dismissed.

The process of shaping drone behavior based on its actual capabilities was taken to extremes, with entire functionality typically used on drones intentionally excluded from the flight control loop (such as the processing of inputs from the altitude sensor or from the flow sensor). The code in the **main** function of Appendix A used a workaround to exclude both sensors from the processing of the extended Kalmann filter used for flight control [55]. These sensors are normally used to achieve better flight stability. For the

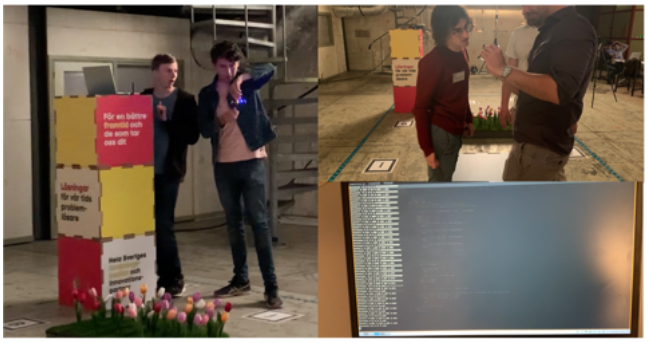

Figure 5: Left: Pilots probe the drone through movement to find out its capabilities and limits. Right: Two pilots examine a drone while sensor input appears on the screen.

challenge, however, four teams out of five decided to use these sensors to command the drone in the vertical direction using their gestures, and eventually gave up on doubling these inputs for the flight control loop, considering this to be too complex to achieve.

Participants looked for other approaches that offered greater simplicity and robustness against the volatility and inaccuracies of the onboard sensors. From pretending that distance readings are millimeter-accurate, for example, the teams realised that the ranges they should work with are rather tens of centimeters. The final program of Appendix A determines drone reactions to the pilot's gestures based on multiple hard-wired parameters, likely obtained through trial and error. These observations apply to the programs of three out of the five participating teams (*Bellman Brothers, Flying Ferrets*, and *RoboNerds*). Rather than shaping the drone behavior programmatically, two teams, *Robo Geeks* and *Spark Speed*, chose to focus on shaping their own bodily movements, instead.

5.2 Exploring Drone Capabilities

Once the participants realised that the HDI seen in the opening talk was not attainable out of the box, they started a process of exploring the drone's capabilities to find out what was actually feasible. This spanned both the drone's hardware and software. As one participant described, it involved collaborative experimentation with different parameters where one team member was in charge of tweaking the parameters by the computer while another interacted with the drone: *"A big part of the problem is just tuning the parameters: the velocity, tuning the push distance, and so on... So we have to try and see is it too fast now? Is it too slow? Is it... Do we need to move too close to it to make it react? So we just try different parameters and see how well it works and our rules is, we don't have the rules, but Leif deals mostly with the computer, he changed the parameters. I'm mostly with the drone, setting it up and moving with it."*

Throughout the event, teams were constantly experimenting in the arenas: changing one parameter at a time and verifying the effect of those changes on the drone behavior. This required plenty of touching and bodily adjustments to the drone, checking connections among the different components, understanding the weak points and what parts are prone to break, plus fixing whatever broke. We observed participants tilting drones, trying to identify its top and bottom, and sometimes hurting their fingers on the propellers in the process. In interviews, participants brought up feelings of concern, both for themselves and for the drone: *"When the drone comes, comes to you from this part [pointing at the torso] I feel more vulnerable, but when it is around a foot, I am not as much vulnerable. When it is around a head or the eye, I feel like I am in danger. But when it is just going around on the ground, I am just worried that the drone itself gets damaged so they cannot fix it"*. The participants also reported on learning to adjust to the drone and developing more and more courage in handling these fragile devices: *"They always look so fragile. So first you are afraid to just handle them: whether you will just destroy it or a sensor will come out. So you have to handle them with a lot of care."* These feelings became essential to participants' developing understandings of the drone capabilities, even though they are not commonly mentioned in papers or drone manuals.

Some teams eventually created *boundary situations* to push the drone to its limits to determine where those are. One team admitted having tried out an approach whereby the drone coped with obstacles by physically bumping into them, making the sensing problem essentially irrelevant as the drone would directly feel the obstacle when it touches it. The team quickly abandoned this idea, facing irremediable breakages at every encounter with an obstacle. In general, crashes became the norm rather than the exception; participants appeared surprised and felt amused whenever things actually (momentarily) worked. Some participants performed several *mock-up flights*. As shown on the left of Fig. 5, team *RoboNerds* entered the arena with the drone in their hands and mimicked an actual run. While doing so, another team member was monitoring on a laptop the sensor readings coming from the drone as it was approaching obstacles or being subject to different piloting gestures. The data was instrumental to understanding the accuracy of the sensors and tuning the different parameters in the code accordingly.

These procedures may be regarded as a crude, bodily approach at solving the visibility problem. Rather than relying on dedicated industry-strength solutions, participants engaged in an intimate form of bodily interaction with the drone to *"get to know it"*, as one of the participants in *Flying Ferrets* put it. These activities – not required by the challenge but ultimately productive for achieving its objectives – unfolded over time, likely because the teams were gaining deeper understandings of the drone and eventually *"got to know it"*, as the same participant declared at the verge of the final day, feeling confident of their outcome.

As the trials progressed, and especially during the actual challenge, some participants started interacting with the drone as if it was a *living entity*, capable of more advanced communication than its actual capability. We observed participants talking to the drone, suggesting where to move, and making gestures akin to giving directions to another person rather than to a robot. While such activities could, of course, be interpreted as directed toward the other people observing the pilot's interaction with the drone – making one's actions and struggles accountable to a human audience – here they came across as genuine attempts at human–drone interaction.

5.3 Training the Drone vs. Training the Pilot

We observed strikingly different approaches at shaping the human–drone interaction necessary to tackle the challenge.

First, adopting an emphasis on *training the drone*, some teams largely concentrated their efforts on producing a program that was sufficiently intelligent to handle sensor inaccuracies and could accurately respond to the pilot gestures. *Bellman Brothers'* code (Appendix A) is one such example, but this approach was taken to an extreme by another team, the *Flying Ferrets*, whose program we report in Appendix B. This team demonstrates how background and prior skills impact the shaping of HDI: members of *Flying Ferrets* were enrolled in an Aerospace Engineering program and had solid knowledge in flight control, including the use of Kalman filters [66] for state estimation.

Flying Ferrets was the only team who wrote their code from scratch, instead of adapting one of the examples provided during the tutorial. Their parameter setting and velocity estimations, as implemented in function **compute_velocity**, demonstrate their intimate knowledge of where and how input data is gathered and of the sensitivity of velocity estimations. The team explicitly defined parameters to strike a trade-off between maximum velocity and accuracy of control in their code, as in **ActionLimit** and **VelocityLimit**. *Flying Ferrets* were also an example of *defensive programming* [51], which they used to handle sensor inaccuracies. Defensive programming is the creation of programs designed to avoid problematic issues before they arise. One common method for achieving this is through code that is meant to deal with any possible scenario thrown at it, making the program able to run properly even through unforeseen situations.

Defensive programming often relies on considering situations whose concrete occurrence might not ever take place.[3] We found evidence of defensive programming, for example, in the change of the default reading frequency of sensors, which many teams applied up to a 5x increase compared to the factory configuration. Obtaining more data allowed these teams to apply aggressive averaging and filtering to exclude outliers. Team *RoboNerds* implemented buffers of up to 100 readings to do so, which is 10 times more than what is normally performed on most existing flight controllers [7]. This increases the drone stability during flights, in exchange of slightly higher energy consumption. The latter, however, was not an issue during the challenge, as the battery lifetime far exceeded the duration of a single run and the technical support made plenty of spare batteries available. Most teams also capped the maximum drone velocity down to half of what the Crazyflie can do, in an attempt to make things easier for the human pilot.

Second, at the opposite extreme, we find teams focused on *training the pilot*. These teams made very limited changes to the example code provided during the tutorials and, instead, spent more time in the arenas to gain the necessary piloting skills. One team, *Robo Geeks*, only added 7 lines of code to one of the tutorial examples and spent most of their time in the arenas running that program, learning how to best command the drone with their gestures.

According to our recordings of the event, *Robo Geeks* spent roughly 40% more time in the arenas than any other team. During the actual challenge, *Robo Geeks* were inadvertently penalised for this approach – which held a lot of promise throughout the trials – because none of the code examples we provided was sufficient to reach all markers. The marker placed on one of the obstacles, in particular, required changing flight altitude, which was not possible with the example code alone. *Robo Geeks* recognised the problem on the last day and, after the initial disappointment, decided to simply skip the problematic marker during the challenge, hoping their well-rehearsed piloting would compensate for this shortcoming by allowing them to collect all other markers in a faster tempo.

[3]Existing statistics [12] report that, when adopting this form of programming style, roughly 30% of the produced code is never actually executed in production. In many ways, the concept is akin to that of defensive driving, in that problems are considered before they arise and not because they occurred in the past.

Figure 6: A pilot uses cardboard paddles as an extension of his body to pilot the drone.

As a result of shaping their own movements, some teams eventually discovered new and unconventional ways to pilot the drone. Team *Spark Speed*, for example, realised that in addition to the two inputs provided with their hands, further input may be provided by other body parts, especially the belly. This allowed them to better control the drone as it could be piloted through one additional input. Another team, *Cyber Ravens*, came to think that their own hands were one source of sensor inaccuracies. This has some technical foundation: the beam of the drone sensors is narrow, so extending the sensed surface may provide more accurate and stable readings. To extend the pilot's physical body so as to provide better inputs to the drone, *Cyber Ravens* eventually opted for wearing self-made cardboard paddles to command the drone (see Fig. 6), to extend the area that the drone sensor could detect.

Most pilots also performed atypical, stilted movements to cope with the localization system's need for line of sight between the base stations and the drone. Some of these put significant strain on the pilot's body (see Fig. 7 for an example). We observed participants bending their backs to stay in the vicinity of the drone with their hands and provide continuous piloting input without standing in the way of the localization system. During the trials, one participant attempted piloting the drone on his knees, but this turned out too impractical and unsafe as the drone propellers ended up on level with the participant's eyes.

In the end, team *Spark Speed* won the challenge, striking the best trade-off between shaping the drone behavior and extending their own capabilities by using not just their hands, but other parts of the body as piloting input. Team *Cyber Ravens* ranked second, thanks in part to their use of cardboard paddles as an extension of the pilot's hands. *Flying Ferrets*, with their accurate tuning of state estimation algorithms, ranked third.

Figure 7: A pilot moves in a stilted manner to maintain line of sight between Lighthouse basestation and drone.

6 SOMATIC ENGAGEMENT IN THE DESIGN OF HUMAN–DRONE INTERACTION: WHAT IS NEEDED?

We acknowledge that our drone challenge is an edge case in that we look at a group of people who are given limited time to get to know the drone and shape their interaction with it. Our call for action, here, is based not only on participants' experiences, but also on the drone experts in our team, as well as our prior work in the area [16, 39, 41–43, 50]. Our research is geared to understand human–drone interaction so as to better support it, with a particular focus on *design-through-use*, as opposed to the traditional design for intended use (see [53, p.79-80]). This approach to design recognises that technology is rarely used exactly as intended, but is, instead, adapted (or subverted) by users to fit their unique context. This is reflected in our data – for example, in how each team approached the challenge differently, some adapting their bodies, others the code, and others a mixture of both. This also resonates with the broad engagement with drones we have seen in a variety of designerly settings [16, 24, 37, 39, 41] - some of which we will draw upon here.

Design-through-use is a way of, not only understanding human–drone interaction, but to generate new ways of conceiving it. As drones move into domestic and social contexts, new ways of engaging or understanding them – ones suitable for non-engineers and technical novices – are warranted. We need to make space for people to actively shape technology into their lives, including helping designers to situate them into their creative processes. This is why we deem our drone challenge with 'novices' a generative research setting, similar to Gamboa's research on children engaging with drones [22]: it sheds light on the inseparable coupling between perception and action when controlling a drone with bodily inputs, and on what is needed to support explorations that result in novel interactions through the mutual shaping of pilots and drones. As we argue next, getting to know drones *aesthetically* is a deeper and more involved process than merely learning to understand how to program them, how to handle their broken propellers, or how to figure out the sensor systems within them.

6.1 Beyond Opening The Black Box

Messy details of interacting with the drone were at the core of the challenge. The team work mostly consisted of failures: crashes and breakdowns led to better understanding of the drone, fostering *bodily knowledge* on how to handle the drone, how to optimise the code, or how to make sense of sensor readings. This is exemplified by how Spark Speed discovered – through trial and error – that different body parts offered more reliable inputs, helping them in understanding the drone's onboard sensors. Handling failure, therefore, occupied the teams for most of the three days of the event - only the team runs during the challenge finale itself went, for the most part, smoothly. That said, even the final performances of the winning team were, at times, interrupted by technical problems. These constant problems and breakdowns are in stark contrast to the *narrative of success* [31] that we often see in design reports. However, as we argue here, and as has been highlighted in prior work [3], not only are real-world robots far from perfect, but breakdowns are, in fact, essential for design practice. They can help us develop technical qualities, articulate mutual understanding within a design team, and allow the team to explore and exploit affordances of the technology at hand [50] – in this case, understanding the complexities of drones. As Fdili-Alaoui argues on the tensions between technology constraints and artistic aims [18, p. 1204]: "*Some of the readers might think that this paper draws a dark portrait of technologies in art, how can we fix this? There is no problem here, there is no solution neither. Artists are experimenting with technology, facing its resistances, pushing its limits.*" We, therefore, echo calls to refrain from binary dichotomies between successes and failures [31].

Considering the teams' explorations to get to know the drones, we note that the challenge involved not so much coding – although some teams chose to write their own – as tweaking parameters: changes were often not algorithmic but adjustments to improve how the drone's sensing picked up on the pilot's movements. The challenge, then, could be framed as changing, rather than shaping, the drone. Importantly, we have illustrated numerous situations

where what is trained and shaped is not the drone but the pilot. As reported above, one of the teams decided to focus almost entirely on training their pilot, adjusting their program only minimally. This approach of focusing on training the pilot held a lot of promise throughout the trials, even though it ultimately introduced some challenges, in particular when the obstacle course was changed ahead of the challenge runs in a way that would have necessitated the drone to have capabilities that it had not been coded to have.

We also saw many instances of pilots making bodily movements to figure out how the drone would respond: while the pilot was moving with the drone, another team member looked at the screen, trying to follow what data is generated and when exactly something goes wrong. It is, after all, impossible to move with the drone while also looking at a laptop screen. This begs the question whether we could have given the teams better tools to explore drone behaviour. We note a fundamental gap between coding the drone behaviour and the drone's behavior once the program runs. The issue largely stems from the *visibility problem* [44] we discussed in Sec. 5.1: much happens "inside" the drone that is not visible from the "outside". Since embedded programming environments for drones are still rather rudimentary, drones often produce incomprehensible behaviors. As Ajaykumar and colleagues found in a survey on end-user programming of robots [2], for the most part, the robot remains a black box. While the survey focused on "end-user program specification" – in essence, tools designed to democratise the complexities of programming robots – our work, while in part similarly motivated, has a different emphasis: Here, the design iterations and mastering of the interactions relied heavily on the *felt* experience of the participants whilst they were directly controlling the drones. We observed that, through "feeling out" and observing what kind of movements resulted in the most stable connection between the drone and the lighthouse, our pilots developed atypical and stilted movements. This somatic aspect is largely missing from the review article [2], prompting us to call for its incorporation into the programming environment. While more can be done to create better embedded programming environments [47] – simulators, visualisations, proper debugging, and ways of watching live data streams – the issues we uncovered here also point to an underlying problem relating to *somatics and bodily knowledge*.

6.2 Perception–Action Loops

As Suchman [60] argues, to act intelligently in a situation, machines need to become *readable/writable*. A pre-requisite for this is that their inner states must become *visible*. Machines must convey their inner state, making them *readable*, but they should also provide us with *inscribable surfaces* that allow us to dynamically change our plans and behaviours to attain our goals depending on, and in response to, what the machine does [60]. We acknowledge that achieving this within the limited computing and energy envelop of nano-drones, usually equipped with low-power resource-constrained microcontrollers, is incredibly challenging. Yet, we argue that to understand drones, we cannot focus solely on watching their behaviour, nor on figuring out how they sense human movement. We need to enter a *perception–action loop* where we can both probe and change the drone's behaviours in a tight loop.

To better explain what we mean by *perception–action loops*, we draw on the evolution of personal computing and an analogy to direct-manipulation. As discussed by Mueller [48], an analogue between early compilers and today's modern interactive compilation techniques can be drawn to modern technologies, such as 3D-printers – or, in our case, drone behaviours. Instead of first sending a blueprint to the 3D-printer and then waiting until the object is drawn, Mueller imagines interactive, direct-manipulated, printers where you can, on the fly, modify the printing process.

One of Mueller's systems, FormFab, allows for such direct control of a 3D-printer, specifically, of the vacuum forming process which is typically a high-temperature moulding process that only allows for one shape to be explored at a time. FormFab uses a work piece that, when warmed up, becomes compliant and can be reshaped. To reshape it, users direct-manipulate a pneumatic system interactively. As they do so, they see the shape in the 3D-printer change in real-time. FormFab allows the user to select a section of the work piece to be heated by a robotic arm. This section, then, becomes compliant and can be reshaped by manipulating a pneumatic system interactively. As users interact with the system, they see the plastic sheet change in real time as a result of the pressure changes.

Löwgren asks us to refine the design of such systems until the experience becomes *pliable* [45]. He argues that a pliable interaction feels like a tight loop between eye and hand, between action and response. For example, when zooming on a map, the pinch gesture makes us feel as if we are inside the map, a very different experience than when using a control panel on the side of the map. The pinch gesture allows for a tight coupling between finger movement and the map. This renders immediate pleasure and a sense of involvement. The objects inside the system become tangible and real to us. Löwgren [45, p. 86] argues that "*[p]liability is a sensuous quality, having to do with how it feels to use the artifact in the here-and-now of the use situation, and as such it plays a role in understanding the aesthetics of interaction.*" It goes beyond function, providing for a somatic aesthetics of sort.

Our drone challenge participants found roundabout ways of creating for those perception–action loops. As discussed in section 5.2, one team placed one team member in front of the computer while another gestured around the drone, so that they, together, could see what sensor data would be rendered depending on the settings in the code. Another team figured out that wearing a cardboard-paddle would enhance the feeling of directing the drone as the readings became less flaky, thereby fostering a stronger sense of bodily connection with the drone. In both cases, teams were seeking a stronger somatic feel for the sensing of the drone.

While it can be argued that these are novice behaviours, these experiences rhyme with our own struggles in creating human–drone interactions and dancing with drones. Note, however, that we are not advocating for the willfulness of the drones to be entirely removed or for aiming at interactions that are always smooth and flawless. Based on our design process for drones on the opera stage [16], we argue that some forms of resistance or glitches in a system can be productive in shaping the aesthetics of the performance. When the choreographer of the opera-drones interacted with them, she was keen to explore those cracks and problems, but they first had to be made felt and somatically known to her. A slow process

of continuously exploring and feeling, through movement, interleaved with re-programming the drones, until she and "*the drones' technological bodies finally came to a state where she could adjust and coordinate her responses in a sort of pre-reflective state. This intercorporeal connection is not something that just arose 'naturally'. It was a learning process that took time. Åsa had to learn to understand the drones' 'otherness' and ability to follow her, a process that required kinesthetic engagement.*" [16, p. 5]

In summary, we argue for translating the pliability quality into interaction tools for drones, engaging users in tight action–perception loops that let them somatically feel (and appreciate) the drone actions – wilful and annoying as they may be.

6.3 It Is Not Natural: Mutual Movement-based Shaping

As Norman [49] points out, designing for some 'natural movement' will not help as meaning-making arises from seeing the consequences – the *feedback* – of a gesture, or movement, as it unfolds in real-time. This feedback needs to be overtly accessible throughout the whole interaction. It is an interaction that is learnt, but that learning can only happen if we can make sense of the interaction. This becomes particularly difficult when we make gestures at a system and cannot disambiguate whether the gestures were done incorrectly or perhaps were not 'seen' by the system. Norman [49, p. 6-7] frames the problem as: *"More important, gestures lack critical clues deemed essential for successful human-computer interaction. Because gestures are ephemeral, they do not leave behind any record of their path, which means that if one makes a gesture and either gets no response or the wrong response, there is little information available to help understand why. The requisite feedback is lacking"*. Drawing on these ideas, programming environments for drones could be of substantial help, if they could not only make those action–feedback loops somatically felt in the moment and as they unfold over time, but also provide relevant feedback of whether the drone 'recognised' a gesture, noted a gesture but could not decode it, or did not even 'see' the pilot. Performing an action with the drone without knowing for sure if it was perceived by the drone makes the learning process very complex.

Crucially, though, issues with drones do not arise solely from a lack of accurate real-time feedback from the software execution in the drone. The material body of the drone itself is fragile and error-prone. In our challenge, we noted repeatedly how teams were mystified as to why their drone acted the way it did. Was it the battery failing? A broken propeller? A sensor malfunction? Teams had to tease out the fragilities of the drone's physical body from other problems, such as a bug in their code, or anomalies, like when the pilot gets in the way of the positioning system. Telling the different potential sources of trouble apart was far from trivial. The messiness of the design process the teams experienced does not dismiss but rather supports, our call for programming environments that elucidate action–feedback loops for somatic appreciation.

Movement-based programming systems [19, 27] are gaining popularity as designers increasingly work with complex algorithms they have not been trained for [6, 68]. We see the somatic training of designers who interact with these movement-based systems as part of a multi-faceted approach to the somatic processes and bodily understanding required to utilise these systems to greater effect. We argue that systems purporting to elicit supposedly 'natural' gestures or movements [8] often assume that such bodily knowledge does not need to be 'trained' or 'developed' in the same way as technical skills, such as coding. However, our study suggests that the cultivation and development of somatic capabilities also plays a fundamental role when developing movement-based robotic systems. A similar argument is made by La Delfa and colleagues [42] who show how participants can develop strong understandings of the inner workings of a drone, given that the drone provides enough somatic signs and signals conveying its state.

Similarly as regular programming systems evolved from textual representations of the application logic to graphical ones, our call for action here is to push programming environments for drones and the like to *elucidate perception–action loops* through somatic engagement. The aim should be to support people's engagement in these loops rather than forcing them to break out of it which is what happened during the challenge when participants needed to go back-and-forth between checking the code on a screen and piloting the drone. We envision augmenting human–drone interaction design processes by making space for somatic sense-making of autonomous robot technologies [25, 26], as this would serve designers in their efforts to uncover the various possibilities – both creative and technological – offered by such technologies [18].

7 EXAMPLES OF DESIGNING FOR PERCEPTION–ACTION LOOPS

Myriads of possible designs can help make the black box of the drone engage in an open-ended perception–action loop with its pilot. Here, we will provide a few examples on how it can be done to make it easier to grasp our conceptual discussion. Our purpose is not to provide an ultimate solution, as we do not consider there to be any solution that would work for every context. Furthermore, the unexpected breakdowns, crashes, and the wilfulness of the drone are, as we argued above, not just a problem to be (partially) fixed, but also an opportunity for aesthetic expression.

We organise the section in line with three themes. In addition to drawing on design work our team members have created for other contexts (see in particular [16, 39, 42, 43]), some of the drone behaviours we discuss are ones we implemented for a second drone challenge that we arranged a year after the one described above.[4]

7.1 Readable/Writeable Drones

As shown in our account of the drone challenge above, it is not always clear to the pilot when the drone is listening to them, or when it is lost, or doing its own thing. Here, we use an example by La Delfa and colleagues – the *How to Train your Drone* (HTTYD) system [39, 42, 43] – to illustrate how a drone can communicate, in an embodied manner, when it is 'looking for' and possibly locates the pilot.

In HTTYD, the participant wears a tracker on each hand (see Fig. 8). These trackers form a triangle space with the drone. Until the

[4]Due to space limitations, we will not go into details of this second drone challenge but, in short, we asked participants to come and dance with a drone. A few trained dancers signed up but most were amateurs. The participants were not asked to program the drone at all, but, instead, to get to know it, explore its expressiveness, and, ultimately, choreograph and prepare a performance.

Figure 8: How to Train your Drone – a person wears trackers on their hands while controlling the drone that hovers near his head.

drone finds the pilot, forming the triangle with the hand trackers, the drone spins in place, emitting a clicking sound. Once it finds the position of the pilot's two hands, the spinning slows down, and the clicking sound becomes progressively faster, coalescing into a rising tone. When the position of the two hands is successfully determined, the spinning comes to an abrupt stop, the drone hovers in place, and turns to face the participant. The rising tone is cut. A singular alien 'eye' on the drone looks at the pilot. This signals that the drone is now ready to move with the pilot (and in fact, learns the dimensions of the current triangle formed between them).

This spinning behaviour reveals the drone's 'inner state', telling the pilot when the drone is attending to them. This makes the drone *readable*. Pilots of HTTYD learned how to wait, keeping still, until the drone eventually detected them. Only then was the drone available to learning a new position, opening its *inscribable surface*. In the HTTYD system, the drone learns a new relative position of the hands this way, and can, as a result, start moving with the pilot, becoming *writable*.

The HTTYD example illustrates one way of portraying the inner workings of a drone in a manner that neither makes the drone into something that it is not (anthropomorphising), nor tries to portray its inner workings "truthfully", in a one-to-one-manner. Not all black boxes need to be opened. Instead, La Delfa designed an interaction we can somatically, kinaesthetically, perceive, learn and act on – even letting us change the drone's behaviour over time.

7.2 Mutual Shaping: A Movement-based Learning Process

Building on the insights from the drone challenge we have described above, we implemented a few minimal, basic behaviours for the second drone challenge that, so to say, let the drone become a partner 'in the dance' with the pilot. We put a camera at the front of the drone so that it could 'see' the pilot (who we can also think of as the dancer) and react to their movements.

The drones were programmed to recognise the dancer's body and move with it. As drones accelerate and decelerate in their own manner, with slight delays compared to the human dancer, wobbling at times, this 'moving with' did not feel or look like a straight line between dancer and drone, but more of a mutual leading and following. The dancers had to learn to adjust to the tempo and movement patterns of the drone. When the dancer was not moving (or the drone failed to 'see' the dancer), the drone would 'dance' on its own, circling around the space, moving up/down, left/right. If the dancer got too close, the drone would move 'around' the dancer (or other obstacles), in order to avoid crashing. This helped to avoid breaking the illusion of a flow between the two.

To engage with these behaviours, the dancer, at times, had to let the drone complete its own (autonomous) movement. If the dancer got in the way of its behaviour right when it was making, for example, a circle around the room, the drone would have to compensate for the disruption – at times crashing, at other times wobbling, stopping, or hovering. The dancer had to learn to see the drone more as a real dance partner with its own agency, letting it complete its dance before directing it to move in some particular direction. Once the dancer knew what to expect (in an embodied, pre-reflective sense), they could shape the aesthetic expression within the limits of these simple drone behaviours. Here, dancers have to adjust their gestures to the drone's algorithms, changing their movements in order to dance *with* the drone. In other words, dancers' movements are shaped by the drone's behaviours.

As a second example of shaping and being shaped by the drone behaviour, let us return to the opera drones. We discussed above how the choreographer, Åsa, came to a somatic understanding, a pre-reflective kinaesthetic meaning-making process [16]. This intercorporeal connection is not something that just arose 'naturally'. It was a learning process that took time. Åsa had to learn to understand the drone's 'otherness' and ability to follow her – a process that required repeated somatic engagement. Her learning process was interleaved with re-programming the drone based on her feedback. For example, the algorithm for following Åsa's hand, moving with her movement, was made so that Åsa could move at the right speed to create a "bent" circular gesture, rather than going straight from A to B. This opened up the drone behaviour to Åsa so that she could start gesturing to it – "go there". The one-year process of mutual shaping between Åsa and the drone became a fertile ground for the aesthetic expression of the resulting opera performance.

These two examples illustrate ways of opening the drone design so that meaning-making can arise between drone and dancer/pilot in a somatically felt sense.

7.3 Making Sense of the Willful Drone: Disambiguating Crashes

We will never fully avoid drone crashes or other unwanted drone behaviours. Instead, we believe the aim should be to help the pilot disambiguate the reasons for crashes and other problems, making it easier to understand what went wrong and why. For example, a problem in the first drone challenge described above arose when the pilot got in the way of the beam of the lighthouse (see Fig 7), causing loss of orientation and potential crashes. This issue was not immediately clear to participants who were new to the drone setup, and it took them time to understand why their drones kept crashing. For the second challenge, we decided to replace the lighthouse solution through adding a camera to the bottom of the

drone. The camera allowed the drone to look down at the floor, letting it stabilise itself by keeping its distance from the ground. By drawing a line on the floor around the performance space, and letting the camera identify this delimited space, the drone could keep away from the edges of the performance scene. Not only did this reduce the number of crashes, the behaviour of the drone also became somewhat more predictable.

Note that if dancers put their hand or foot underneath the drone, it lost its bearings as the limb got in the way of 'seeing' the floor, moving in unpredictable ways. This might sound like a bad solution – and there are surely other ways to design the sensing of location that could be better – but the advantage lies in the clarity of the behaviour. It is easier to figure out what is happening if the action–perception loop is tight. Putting a foot under the drone immediately makes it 'go crazy', while putting one's body in-between the drone and the lighthouse did not render equally clear feedback.

Rather than assuming that we can design perfect drone behaviours, avoiding crashes altogether, we propose acknowledging that crashes and other unintended behaviours will happen. We cannot predict every action that a pilot might do, but once we see them, we can note whether the relationship between action and what is perceived is clear – step by step modifying the drone behaviour until reasons for crashes are easier to disambiguate for the pilot. In this example, dancers might even find this unpredictable behaviour aesthetically interesting and choose to integrate it into their performance.

8 CONCLUSION

We contribute an empirical, somaesthetically focused account of current challenges in how to shape human–drone interaction. Reporting on a three-day drone challenge, we have illustrated how teams *1)* shifted from aiming for seamless human–drone interaction, to seeing drones as fragile, wilful, and prone to crashes; *2)* engaged in intimate, bodily interactions to more precisely understand, probe, and delimit their drone's capabilities; and *3)* adopted different strategies, emphasising either training the drone or training the pilot in a somatic, movement-based manner.

Drawing upon insights from the drone challenge, a second challenge we organised a year later, as well as further drone design projects our team members have worked on, we argue for supporting somatic engagement in the design and development of drones and drone tools to: elucidate and enable tighter *perception–action loops*; rely on somatic signs and signals to let pilots *feel* the drone's inner states rather than having to reason about them; and, at times, make unwanted drone behaviours, such as the drone losing track of where it is, be triggered by movements that pilots can more readily make sense of. Embedded programming tools for drones do not only need to allow programmers to adequately change the code or sensor filters of the drones, but also to 'train' their pilots through providing somatic signs and signals that can be probed and felt in real-time [41]. This will let pilots learn about how to change their own movements to fly the drone, until they create a new way of moving together with the drone. It will be in this interplay, between shaping and being shaped by the drone – perceiving and acting on what is perceived through all the senses – that the aesthetic exploration of human–drone interaction will thrive.

ACKNOWLEDGMENTS

We thank everyone who contributed to the two challenges at the Digital Futures Drone Arena, especially all participants, Fatemeh Bakhshoudeh and Christoffer Eriksson, as well as the staff at Digital Futures. This work is supported by The Digital Futures Drone Arena, a Digital Futures Demonstrator Project at the Department of Computer and Systems Sciences at Stockholm University and The Connected Intelligence Unit at Research Institutes of Sweden (RISE), and the Wallenberg AI, Autonomous Systems and Software Program – Humanity and Society (WASP-HS) through a Marianne and Marcus Wallenberg Foundation project MMW 2019.0228.

REFERENCES

[1] Mikhail Afanasov, Alessandro Djordjevic, Feng Lui, and Luca Mottola. 2019. FlyZone: A Testbed for Experimenting with Aerial Drone Applications. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 67–78. https://doi.org/10.1145/3307334.3326106

[2] Gopika Ajaykumar, Maureen Steele, and Chien-Ming Huang. 2021. A Survey on End-User Robot Programming. *ACM Comput. Surv.* 54, 8, Article 164 (October 2021), 36 pages. https://doi.org/10.1145/3466819

[3] Patrícia Alves-Oliveira, Maria Luce Lupetti, Michal Luria, Diana Löffler, Mafalda Gamboa, Lea Albaugh, Waki Kamino, Anastasia K. Ostrowski, David Puljiz, Pedro Reynolds-Cuéllar, Marcus Scheunemann, Michael Suguitan, and Dan Lockton. 2021. Collection of Metaphors for Human-Robot Interaction. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference* (Virtual Event, USA) *(DIS '21)*. Association for Computing Machinery, New York, NY, USA, 1366–1379. https://doi.org/10.1145/3461778.3462060

[4] Birgir Baldursson, Tim Björk, Lisa Johansson, Agnes Rickardsson, Ellen Widerstrand, Mafalda Gamboa, and Mohammad Obaid. 2021. DroRun: Drone Visual Interactions to Mediate a Running Group. In *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction* (Boulder, CO, USA) *(HRI '21 Companion)*. Association for Computing Machinery, New York, NY, USA, 148–152. https://doi.org/10.1145/3434074.3447148

[5] Mehmet Aydin Baytas, Damla Çay, Yuchong Zhang, Mohammad Obaid, Asim Evren Yantaç, and Morten Fjeld. 2019. The Design of Social Drones: A Review of Studies on Autonomous Flyers in Inhabited Environments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300480

[6] Cristian Bogdan, Vasiliki Tsaknaki, Charles Windlin, Marianela Ciolfi Felice, Ozgun Kilic Afsar, Sara Eriksson, Ylva Fernaeus, and Pedro Sanches. 2020. Programming for Moving Bodies. In *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society* (Tallinn, Estonia) *(NordiCHI '20)*. Association for Computing Machinery, New York, NY, USA, Article 132, 3 pages. https://doi.org/10.1145/3419249.3420069

[7] Endri Bregu, Nicola Casamassima, Daniel Cantoni, Luca Mottola, and Kamin Whitehouse. 2016. Reactive Control of Autonomous Drones. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services* (Singapore, Singapore) *(MobiSys '16)*. Association for Computing Machinery, New York, NY, USA, 207–219. https://doi.org/10.1145/2906388.2906410

[8] Jessica R. Cauchard, Jane L. E, Kevin Y. Zhai, and James A. Landay. 2015. Drone and Me: An Exploration into Natural Human-Drone Interaction. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Osaka, Japan) *(UbiComp '15)*. Association for Computing Machinery, New York, NY, USA, 361–365. https://doi.org/10.1145/2750858.2805823

[9] Jessica R. Cauchard, Jane L. E, Kevin Y. Zhai, and James A. Landay. 2015. Drone and Me: An Exploration into Natural Human-Drone Interaction. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Osaka, Japan) *(UbiComp '15)*. Association for Computing Machinery, New York, NY, USA, 361–365. https://doi.org/10.1145/2750858.2805823

[10] Jessica R. Cauchard, Mohamed Khamis, Jérémie Garcia, Matjaž Kljun, and Anke M. Brock. 2021. Toward a roadmap for human-drone interaction. *Interactions* 28, 2 (March 2021), 76–81. https://doi.org/10.1145/3447889

[11] Jessica Rebecca Cauchard, Kevin Y. Zhai, Marco Spadafora, and James A. Landay. 2016. Emotion Encoding in Human-Drone Interaction. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction (HRI '16)*. IEEE Press, Christchurch, New Zealand, 263–270.

[12] Vianney Côté, Pierre Bourque, Serge Oligny, and N. Rivard. 1988. Software metrics: An overview of recent results. *Journal of Systems and Software* 8, 2 (1988), 121–131. https://doi.org/10.1016/0164-1212(88)90005-2

[13] Vinciane Despret. 2013. Responding Bodies and Partial Affinities in Human–Animal Worlds. *Theory, Culture & Society* 30, 7–8 (2013), 51–76. https://doi.org/10.1177/0263276413496852

[14] Paul Dourish. 2001. *Where the action is.* MIT press, Cambridge, US.

[15] Sara Eriksson, Kristina Höök, Richard Shusterman, Dag Svanes, Carl Unander-Scharin, and Åsa Unander-Scharin. 2020. Ethics in Movement: Shaping and Being Shaped in Human-Drone Interaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376678

[16] Sara Eriksson, Åsa Unander-Scharin, Vincent Trichon, Carl Unander-Scharin, Hedvig Kjellström, and Kristina Höök. 2019. Dancing With Drones: Crafting Novel Artistic Expressions Through Intercorporeality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland, UK) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300847

[17] Michael Fagan. 2002. *Design and Code Inspections to Reduce Errors in Program Development.* Springer, Berlin Heidelberg, 575–607. https://doi.org/10.1007/978-3-642-59412-0_35

[18] Sarah Fdili Alaoui. 2019. Making an Interactive Dance Piece: Tensions in Integrating Technology in Art. In *Proceedings of the 2019 on Designing Interactive Systems Conference* (San Diego, CA, USA) *(DIS '19)*. Association for Computing Machinery, New York, NY, USA, 1195–1208. https://doi.org/10.1145/3322276.3322289

[19] Rebecca Fiebrink and Perry R Cook. 2010. The Wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)*, Vol. 3. Citeseer, Utrecht, 2–1.

[20] Dario Floreano and Robert J Wood. 2015. Science, technology and the future of small autonomous drones. *Nature* 521, 7553 (2015), 460–466. https://doi.org/10.1038/nature14542

[21] Markus Funk. 2018. Human-Drone Interaction: Let's Get Ready for Flying User Interfaces! *Interactions* 25, 3 (April 2018), 78–81. https://doi.org/10.1145/3194317

[22] Mafalda Gamboa. 2022. Living with Drones, Robots, and Young Children: Informing Research through Design with Autoethnography. In *Nordic Human-Computer Interaction Conference* (Aarhus, Denmark) *(NordiCHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 27, 14 pages. https://doi.org/10.1145/3546155.3546658

[23] Mafalda Gamboa, Mehmet Aydın Baytaş, Sjoerd Hendriks, and Sara Ljungblad. 2023. Wisp: Drones as Companions for Breathing. In *Proceedings of the Seventeenth International Conference on Tangible, Embedded, and Embodied Interaction* (Warsaw, Poland) *(TEI '23)*. Association for Computing Machinery, New York, NY, USA, Article 9, 16 pages. https://doi.org/10.1145/3569009.3572740

[24] Mafalda Gamboa, Mohammad Obaid, and Sara Ljungblad. 2021. Ritual Drones: Designing and Studying Critical Flying Companions. In *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction* (Boulder, CO, USA) *(HRI '21 Companion)*. Association for Computing Machinery, New York, NY, USA, 562–564. https://doi.org/10.1145/3434074.3446363

[25] Maliheh Ghajargar, Jeffrey Bardzell, Alison Smith Renner, Peter Gall Krogh, Kristina Höök, David Cuartielles, Laurens Boer, and Mikael Wiberg. 2021. From "Explainable AI" to "Graspable AI". In *Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction* (Salzburg, Austria) *(TEI '21)*. Association for Computing Machinery, New York, NY, USA, Article 69, 4 pages. https://doi.org/10.1145/3430524.3442704

[26] Maliheh Ghajargar, Jeffrey Bardzell, Alison Marie Smith-Renner, Kristina Höök, and Peter Gall Krogh. 2022. Graspable AI: Physical Forms as Explanation Modality for Explainable AI. In *Proceedings of the Sixteenth International Conference on Tangible, Embedded, and Embodied Interaction* (Daejeon, Republic of Korea) *(TEI '22)*. Association for Computing Machinery, New York, NY, USA, Article 53, 4 pages. https://doi.org/10.1145/3490149.3503666

[27] Marco Gillies. 2019. Understanding the Role of Interactive Machine Learning in Movement Interaction Design. *ACM Trans. Comput.-Hum. Interact.* 26, 1, Article 5 (February 2019), 34 pages. https://doi.org/10.1145/3287307

[28] Viviane Herdel, Anastasia Kuzminykh, Andrea Hildebrandt, and Jessica R. Cauchard. 2021. Drone in Love: Emotional Perception of Facial Expressions on Flying Robots. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 716, 20 pages. https://doi.org/10.1145/3411764.3445495

[29] Julia M. Hildebrand. 2021. *Aerial Play: Drone Medium, Mobility, Communication, and Culture.* Springer, Singepore. https://doi.org/10.1007/978-981-16-2195-6

[30] Masoumehsadat Hosseini, Tjado Ihmels, Ziqian Chen, Marion Koelle, Heiko Müller, and Susanne Boll. 2023. Towards a Consensus Gesture Set: A Survey of Mid-Air Gestures in HCI for Maximized Agreement Across Domains. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 311, 24 pages. https://doi.org/10.1145/3544548.3581420

[31] Noura Howell, Audrey Desjardins, and Sarah Fox. 2021. Cracks in the Success Narrative: Rethinking Failure in Design Research through a Retrospective Trioethnography. *ACM Trans. Comput.-Hum. Interact.* 28, 6, Article 42 (November 2021), 31 pages. https://doi.org/10.1145/3462447

[32] Felix Huppert, Gerold Hoelzl, and Matthias Kranz. 2021. GuideCopter - A Precise Drone-Based Haptic Guidance Interface for Blind or Visually Impaired People. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 218, 14 pages. https://doi.org/10.1145/3411764.3445676

[33] Kristina Höök. 2018. *Designing with the Body: Somaesthetic Interaction Design.* The MIT Press, Cambridge, US. https://doi.org/10.7551/mitpress/11481.001.0001

[34] Julian Kaduk, Müge Cavdan, Knut Drewing, Argiro Vatakis, and Heiko Hamann. 2023. Effects of Human-Swarm Interaction on Subjective Time Perception: Swarm Size and Speed. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction* (Stockholm, Sweden) *(HRI '23)*. Association for Computing Machinery, New York, NY, USA, 456–465. https://doi.org/10.1145/3568162.3578626

[35] Kari Daniel Karjalainen, Anna Elisabeth Sofia Romell, Photchara Ratsamee, Asim Evren Yantac, Morten Fjeld, and Mohammad Obaid. 2017. Social Drone Companion for the Home Environment: a User-Centric Exploration. In *Proceedings of the 5th International Conference on Human Agent Interaction* (Bielefeld, Germany) *(HAI '17)*. Association for Computing Machinery, New York, NY, USA, 89–96. https://doi.org/10.1145/3125739.3125774

[36] Md. Nafiz Hasan Khan and Carman Neustaedter. 2019. An Exploratory Study of the Use of Drones for Assisting Firefighters During Emergency Situations. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland, UK) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300502

[37] Heesoon Kim and James A. Landay. 2018. Aeroquake: Drone Augmented Dance. In *Proceedings of the 2018 Designing Interactive Systems Conference* (Hong Kong, China) *(DIS '18)*. Association for Computing Machinery, New York, NY, USA, 691–701. https://doi.org/10.1145/3196709.3196798

[38] Nina Kov. 2012. *Copter.* Stanford. Retrieved Feb 12, 2023 from http://ccrma.stanford.edu/~jos/bayes/bayes.html

[39] Joseph La Delfa. 2023. *Cultivating Mechanical Sympathy: Making meaning with ambiguous machines.* Ph. D. Dissertation. KTH Royal Institute of Technology. https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-337183

[40] Joseph La Delfa, Mehmet Aydin Baytaş, Emma Luke, Ben Koder, and Florian 'Floyd' Mueller. 2020. Designing Drone Chi: Unpacking the Thinking and Making of Somaesthetic Human-Drone Interaction. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference* (Eindhoven, Netherlands) *(DIS '20)*. Association for Computing Machinery, New York, NY, USA, 575–586. https://doi.org/10.1145/3357236.3395589

[41] Joseph La Delfa, Mehmet Aydin Baytas, Rakesh Patibanda, Hazel Ngari, Rohit Ashok Khot, and Florian 'Floyd' Mueller. 2020. Drone Chi: Somaesthetic Human-Drone Interaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376786

[42] Joseph La Delfa, Rachael Garrett, Airi Lampinen, and Kristina Höök. 2024. Articulating Mechanical Sympathy for Somaesthetic Human–Machine Relations. In *Proceedings of the 2024 ACM Conference on Designing Interactive Systems.* ACM Press, Copenhagen, Denmark, 1–18. https://doi.org/10.1145/3643834.3661514

[43] Joseph La Delfa, Rachael Garrett, Airi Lampinen, and Kristina Höök. 2024. How to Train Your Drone - Exploring the umwelt as a design metaphor for human-drone interaction. In *Proceedings of the 2024 ACM Conference on Designing Interactive Systems.* ACM Press, Copenhagen, Denmark, 1–14. https://doi.org/10.1145/3643834.3660737

[44] Edward Ashford Lee and Sanjit Arunkumar Seshia. 2016. *Introduction to embedded systems: A cyber-physical systems approach.* MIT Press, Cambridge, US.

[45] Jonas Löwgren. 2007. Pliability as an Experiential Quality: Exploring the Aesthetics of Interaction Design. *Artifact* 1, 2 (2007), 85–95. https://doi.org/10.1080/17493460600976165 arXiv:https://doi.org/10.1080/17493460600976165

[46] Andy Miah. 2020. *Drones: The Brilliant, The Bad and The Beautiful.* Emerald Group Publishing, Bradford, Great Britain.

[47] Luca Mottola, Amy L. Murphy, and Gian Pietro Picco. 2006. Pervasive games in a mote-enabled virtual world using tuple space middleware. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games* (Singapore) *(NetGames '06)*. Association for Computing Machinery, New York, NY, USA, 29–es. https://doi.org/10.1145/1230040.1230098

[48] Stefanie Mueller. 2016. *Interacting with Personal Fabrication Devices.* Institutional Repository of the University of Potsdam, Potsdam, Germany. https://doi.org/10.1515/itit-2017-0041

[49] Donald A. Norman. 2010. Natural user interfaces are not natural. *Interactions* 17, 3 (may 2010), 6–10. https://doi.org/10.1145/1744161.1744163

[50] Kristina Popova, Rachael Garrett, Claudia Núñez Pacheco, Airi Lampinen, and Kristina Höök. 2022. Vulnerability as an ethical stance in soma design processes. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*

(New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 178, 13 pages. https://doi.org/10.1145/3491102.3501994

[51] Xiaohu Qie, Ruoming Pang, and Larry Peterson. 2003. Defensive programming: using an annotation toolkit to build DoS-resistant software. *SIGOPS Oper. Syst. Rev.* 36, SI (December 2003), 45–60. https://doi.org/10.1145/844128.844134

[52] Natasha Randall and Selma Sabanovic. 2023. A Picture Might Be Worth a Thousand Words, But It's Not Always Enough to Evaluate Robots. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction* (Stockholm, Sweden) *(HRI '23)*. Association for Computing Machinery, New York, NY, USA, 437–445. https://doi.org/10.1145/3568162.3576970

[53] Johan Redstrom. 2017. *Making design theory*. MIT Press, Cambridge, US.

[54] Matt Rogers. 2012. Contextualizing Theories and Practices of Bricolage Research. *The Qualitative Report* 17, 48 (2012), 1–17. https://doi.org/10.46743/2160-3715/2012.1704

[55] Stergios I. Roumeliotis and George A. Bekey. 2000. Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation.*, Vol. 3. IEEE, San Francisco, CA, USA, 2985–2992.

[56] Maxine Sheets-Johnstone. 1999. Emotion and movement. A beginning empirical-phenomenological analysis of their relationship. *Journal of Consciousness Studies* 6, 11-12 (1999), 259–277. https://www.ingentaconnect.com/content/imp/jcs/1999/00000006/f0020011/1002

[57] Richard Shusterman. 2008. *Body consciousness: A philosophy of mindfulness and somaesthetics*. Cambridge University Press, Cambridge.

[58] Mario Silvagni, Andrea Tonoli, Enrico Zenerino, and Marcello Chiaberge. 2016. Multipurpose UAV for search and rescue operations in mountain avalanche events. *Geomatics, Natural Hazards and Risk* 8, 1 (Oct. 2016), 18–33. https://doi.org/10.1080/19475705.2016.1238852

[59] Mousa Sondoqah, Fehmi Abdesslem, Kristina Popova, Moira McGregor, Jospeh La Delfa, Rachael Garrett, Airi Lampinen, Luca Mottola, and Kristina Höök. 2024. Programming Human-Drone Interactions: Lessons from the Drone Arena Challenge. In *Proceedings of the 1oth International Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DRONET - colocated with ACM MobiSys)*. ACM Press, Tokyo, Japan, 1–10.

[60] Lucy A. Suchman. 1997. *From Interactions to Integrations*. Springer US, Boston, MA, 3–3. https://doi.org/10.1007/978-0-387-35175-9_1

[61] Dante Tezza and Marvin Andujar. 2019. The state-of-the-art of human–drone interaction: A survey. *IEEE Access* 7 (2019), 167438–167454. https://doi.org/10.1109/ACCESS.2019.2953900

[62] Evgeny Tsykunov, Ruslan Agishev, Roman Ibrahimov, Luiza Labazanova, Taha Moriyama, Hiroyuki Kajimoto, and Dzmitry Tsetserukou. 2019. SwarmCloak: Landing of a Swarm of Nano-Quadrotors on Human Arms. In *SIGGRAPH Asia 2019 Emerging Technologies* (Brisbane, QLD, Australia) *(SA '19)*. Association for Computing Machinery, New York, NY, USA, 46–47. https://doi.org/10.1145/3355049.3360542

[63] Jacqueline Urakami and Katie Seaborn. 2023. Nonverbal Cues in Human–Robot Interaction: A Communication Studies Perspective. *J. Hum.-Robot Interact.* 12, 2, Article 22 (mar 2023), 21 pages. https://doi.org/10.1145/3570169

[64] Sanne van Waveren, Rasmus Rudling, Iolanda Leite, Patric Jensfelt, and Christian Pek. 2023. Increasing Perceived Safety in Motion Planning for Human-Drone Interaction. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction* (Stockholm, Sweden) *(HRI '23)*. Association for Computing Machinery, New York, NY, USA, 446–455. https://doi.org/10.1145/3568162.3576966

[65] Dolores R. Wallace and Roger U. Fujii. 1989. Software verification and validation: an overview. *IEEE Software* 6, 3 (May 1989), 10–17. https://doi.org/10.1109/52.28119

[66] Greg Welch and Gary Bishop. 2001. An introduction to the Kalman filter. *Proc of SIGGRAPH, Course* 8, 27599-23175 (2001), 41. https://perso.crans.org/club-krobot/doc/kalman.pdf

[67] Nialah Jenae Wilson-Small, David Goedicke, Kirstin Petersen, and Shiri Azenkot. 2023. A Drone Teacher: Designing Physical Human-Drone Interactions for Movement Instruction. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction* (Stockholm, Sweden) *(HRI '23)*. Association for Computing Machinery, New York, NY, USA, 311–320. https://doi.org/10.1145/3568162.3576985

[68] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 2020. Re-examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376301

A CODE EXAMPLE

An excerpt of code produced by team *Bellman Brothers*. We use **#...** to omit lines of code that are not relevant to the discussion. Because of formatting, we could not keep the regular Python indentation. The whole program is ≈ 400 lines of code.

```
from functools
    import reduce
import time
from cflib.positioning.motion_commander
     import MotionCommander
from Utility.extension.decks.deck
     import DeckType
from utils
     import ActionLimit, VelocityLimit,get_vx, get_vy
from Utility.extension.extended_crazyflie
     import ExtendedCrazyFlie
from Utility.extension.decks.z_ranger
     import ZRanger
import numpy as np
import matplotlib.pyplot as plt

SPEED_CLIP = 1
BATTERY_LIMIT = 10
ADJUST_VELOCITY = 0.3 # [m/s]
threshold = 0.1 # [m]
DEFAULT_HEIGHT = 1
prev = 0 # 0=hovering, -1=lowering, +1=raising
def which_region(d,push_range,pull_range,unsensible_range):
    # ...

def adjust_height(zrange_state : float):
    global prev
    h = zrange_state
    if (h < DEFAULT_HEIGHT + threshold) and prev < 1:
        prev = 1
        # zranger.contribute_to_state_estimate = False
        # disable zrange contribution
        return ADJUST_VELOCITY
    elif (h > DEFAULT_HEIGHT - threshold) and prev >-1:
        prev = -1
        zranger.contribute_to_state_estimate = True
        return -ADJUST_VELOCITY
    elif prev != 0:
        prev = 0
        return 0

class state_to_filter:
    def __init__(self,n_steps):
        self.array = np.zeros((n_steps,))
        self.mean_value = 0
        self.measurement_ready = None

    def add_new_value(self,value):
        self.array = np.hstack((value,self.array[:-2]))
        # mask = (self.array != 0.)
        self.mean_value = np.mean(self.array)
        # self.measurement_ready = self.array[0]!=0
n_avg = 20
front_range = state_to_filter(n_avg)
back_range  = state_to_filter(n_avg)
left_range  = state_to_filter(n_avg)
right_range = state_to_filter(n_avg)

#def get_speed_command(distance,lower_bound,upper_bound):
    # ...

def get_speed_command(distance,lower_bound,upper_bound):
    if distance>lower_bound and distance<upper_bound :
        gap_width = np.abs(upper_bound-lower_bound)
        current_gap_dist = np.abs(distance-lower_bound)
```

```
        return 0.4
    else :
        return 0

def script(multiranger_state : dict, mc : MotionCommander) :
    # ...
          front_range.
             add_new_value(multiranger_state['front']/1000)
          back_range.
             add_new_value( multiranger_state['back'] /1000)
          left_range.
             add_new_value( multiranger_state['right']/1000)
          right_range.
             add_new_value( multiranger_state['left'] /1000)

          v_front =
                - get_speed_command(front_range.mean_value,
                                    push_range_start,
                                         push_range_end)
          v_back =
                get_speed_command(back_range.mean_value,
                                    push_range_start,
                                       push_range_end)
          v_right =
                - get_speed_command(right_range.mean_value,
                                    push_range_start,
                                        push_range_end)
          v_left =
                get_speed_command(left_range.mean_value,
                                    push_range_start,
                                       push_range_end)
        # front_readings.append(front_range.mean_value)
        # left_readings .append(left_range .mean_value)
        # right_readings.append(right_range .mean_value)
        # back_readings .append(back_range.mean_value)
          z_range = ecf.coordination_manager.
                get_observable_state(zranger.
                                    observable_name)['zrange']/1000
          #   z_readings.append(z_range)
          vz = adjust_height(z_range)*0
          mc.start_linear_motion(v_front+v_back,
                                     v_right+v_left,vz ) # raise up
    # ...

if __name__ == '__main__':
        # ...
        # disabling height and/or flow measurament from EKF
        zranger : ZRanger = None
        # disable flow contribution
        if DeckType.bcFlow2 in ecf.decks:
            ecf.decks[DeckType.bcFlow2].
                 contribute_to_state_estimate = True
        # get the ZRanger of the FlowDeck
                 zranger = ecf.decks[DeckType.bcFlow2].zranger
        if DeckType.bcZRanger2 in ecf.decks:
            zranger = ecf.decks[DeckType.bcZRanger2]
            # disable zrange contribution
            zranger.contribute_to_state_estimate = True
```

B CODE EXAMPLE

An excerpt of code produced by teams *Flying Ferrets*. We use **#...** to omit lines of code that are not relevant to the discussion. Because of formatting, we could not keep the regular Python indentation. The whole program is ≈ 380 lines of code.

```
import logging
import sys
import time
# ...
import cflib.crtp  # to scan for Crazyflies instances
# to easily connect/send/receive data from a Crazyflie
from cflib.crazyflie import Crazyflie
# wrapper around the normal Crazyflie class
from cflib.crazyflie.syncCrazyflie import SyncCrazyflie
# to help connecting to a Crazyflie with a URI
from cflib.utils import uri_helper
from Utility.log import Log  # to log position

# to help moving the drone
from cflib.positioning.motion_commander import MotionCommander
# to use the Multiranger deck
from cflib.utils.multiranger import Multiranger

#...

ADJUST_VELOCITY = 0.2 # [m/s]
threshold = 0.1 # [m]
DEFAULT_HEIGHT = 0.2
prev = 0 #
class ActionLimit():
    #measure unit millimeters
    MIN = 0.150
    MAX = 0.400
    SAFE = 0.100

class VelocityLimit():
    #measure unit meters/second
    MIN = 0
    MAX = 0.8

def is_close(range, distance):
    if range is None:
        return False
    else:
        return range < distance

def is_far(range, distance):
    if range is None:
        return False
    else:
        return range > distance

def compute_velocity(value, limits=ActionLimit) -> float:
        #fixing values in the range (0, ACTION_LIMIT)
        value = ActionLimit.MIN if value < ActionLimit.MIN
                                else value
        value = ActionLimit.MAX if value > ActionLimit.MAX
                                else value
        # NewValue = (((OldValue - OldMin) * (NewMax - NewMin))
                                #/ (OldMax - OldMin)) + NewMin where:
            # OldValue = range_in_mm
            # NewValue = velocity_in_ms
            # OldMin = ActionLimit.MAX (range)
            # OldMax = ActionLimit.MIN (range)
            # NewMin = VelocityLimit.MIN (velocity)
            # NewMax = VelocityLimit.MAX (velocity)
        # NOTICE: we inverted the ActionLimits to get
        # the inverted range conversion
        return (((value - ActionLimit.MAX) *
                        (VelocityLimit.MAX - VelocityLimit.MIN)) /
```

```
                        (ActionLimit.MIN - ActionLimit.MAX)) +
                        VelocityLimit.MIN

# this function will compute the Velocity in the x-axis direction
def get_vx(front, back, limits=ActionLimit)-> float:
    vx = 0
    if(ActionLimit.MIN <= back <= ActionLimit.MAX):
        #back gives a push in the positive x direction
        vx += compute_velocity(back)
    if(ActionLimit.MIN <= front <= ActionLimit.MAX):
        #back gives a push in the negative x direction
        vx -= compute_velocity(front)
    return vx
# this function will compute the Velocity in the y-axis direction
def get_vy(right, left, limits=ActionLimit)-> float:
    vy = 0
    if(ActionLimit.MIN <= right <= ActionLimit.MAX):
        #back gives a push in the positive y direction
        vy += compute_velocity(right)
    if(ActionLimit.MIN <= left <= ActionLimit.MAX):
        #back gives a push in the negative y direction
        vy -= compute_velocity(left)
    return vy

def main():
    # Initialize the low-level drivers
    cflib.crtp.init_drivers()

    #...

    cf = Crazyflie(rw_cache='./cache')
    with SyncCrazyflie(URI, cf=cf) as scf:
        print('SyncCrazyflie open')
        # to start logging position (DO NOT REMOVE)
        with Log(scf) as log:

            # ...
            with MotionCommander(scf,
                        default_height=DEFAULT_HEIGHT)
                as motion_commander:
                print('MotionCommander open')
                with Multiranger(scf) as multiranger:
                    print('Multiranger open')
                    keep_flying = True
                    is_started = False
                    velocity_x = 0.0
                    velocity_y = 0.0

                    lastt = time.time()
                    time.sleep(1)
                    while keep_flying:
                        #vz = compute_velocity
                            #(multiranger.down, 10000.0)
                        #print(multiranger.front, multiranger.back,
                            #multiranger.right, multiranger.left)

                        front = multiranger.front
                        back = multiranger.back
                        left = multiranger.left
                        right = multiranger.right
                        if front is None:
                            front = 1000.0
                        if back is None:
                            back = 1000.0
                        if left is None:
                            left = 1000.0
                        if right is None:
                            right = 1000.0
```

```
                        if is_close(multiranger.down,
                                    DEFAULT_HEIGHT - threshold):
                            vz = ADJUST_VELOCITY*2
                            #motion_commander.
                                #start_linear_motion(0, 0,
                                #ADJUST_VELOCITY)
                        elif is_far(multiranger.down,
                                    DEFAULT_HEIGHT + threshold):
                            vz = -ADJUST_VELOCITY
                            #motion_commander.
                                #start_linear_motion(0,0,
                                #-ADJUST_VELOCITY)
                        else:
                            vz = 0
                            #motion_commander.
                            #start_linear_motion(0, 0, 0)

                        if multiranger is None:
                            print("Multiranger is None!")
                            motion_commander.land()
                            time.sleep(0.5)
                            break

                        if time.time()-lastt > 0.1:
                            print(f"{front} {back} {right} {left}")

                        vx = get_vx(front, back)
                        vy = get_vy(right, left)
                        if time.time()-lastt > 0.1:
                          print(f"{mr.down} {vx} {vy} {vz}")
                          lastt = time.time()
                        motion_commander.
                             start_linear_motion(vx, vy, vz)
                        time.sleep(0.01)

if __name__ == "__main__":
    main()
```