SharpFlying

Design and implementation of a generic multi-service framework for autonomous indoor flight & An exploratory study into autonomous indoor Human-Drone Interaction

Master Thesis
hci102f19

Aalborg University
Software

Copyright © Aalborg University 2019

Software
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
SharpFlying

Theme:
Master Thesis in Human-Computer Interaction

Project Period:
Spring 2019

Project Group:
hci102f19

Participant(s):
Kasper Østergaard Helsted
Steffen Darby Carlsen

Supervisor(s):
Mikael Skov

Copies: 3

Page Numbers: 16

Date of Completion:
10-06-2019

Abstract:

In current time, research into the use of drones in terms of autonomy has become more popular. This has caused researchers to attempt and find a singular technology and solution for solving indoor navigation with drones. Even though some of this research has proven to be successful, none of the found research has attempted to combine multiple solutions into a singular framework. Thus, we designed and implemented SharpFlying, a generic and extendable multi-service framework for autonomous indoor drones. In order to validate our framework, we implemented three proof of concept services. These services were tested, individually and together, to get an insight into whether or not a multi-service structure can perform equal to, or greater than a singular solution. Our results show that by combining multiple services, you gain vastly better results compared to a singular service. Our first study showed how to develop autonomous drones, however, research has yet to figure out a set of design guidelines as to how we should interact with drones in an indoor environment. Based on this, we created a 3-parts exploratory study into the world of indoor autonomous Human-Drone Interaction. Our tasks focused on primary interaction during navigation, voice interaction and secondary interaction. Our results show that participants expected the drone to behave under a level of *trust*. Trust was a metaphor expressed by some of the participants, they related this word to the behavior of the drone and how flawless they expected it to be. Our study outlines a series of design insights for future interaction development of indoor autonomous drones.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Preface

This master thesis has been made by hci102f19, 10th semester Software students from Aalborg University. The focus of the master thesis is on indoor autonomous drone flight from the perspectives of development of a multi-service framework and exploration of interaction techniques.

This master thesis consists of two academic papers each detailing their own research in regards to autonomous drone flight.

The master thesis has been written in cooperation with the project supervisor, Mikael Skov, who we express gratitude and thanks towards.

Aalborg University, 11th June 2019.

Kasper Østergaard Helsted
khelst13@student.aau.dk

Steffen Darby Carlsen
sdca14@student.aau.dk

Summary

The purpose of this project is to investigate the world of autonomous drones from a development and interaction perspective. The first article starts by consulting the related work in regards existing solutions within autonomous drone navigation. This gives an insight into how researchers has already attempted to solve the many problems of autonomous navigation, both in regards to indoor and outdoor. This showed that the related work always focused on using one technology or method to perform indoor autonomous navigation. Based on the related work, we extracted multiple approaches to indoor navigation and created the framework, SharpFlying. This framework is a service-focused extendable and generic implementation, meaning that it is created to allow for other developers to continue the development by implementing more services using the base-classes we have implemented.

As a proof of concept, we created three different services: Vision-, Distance- and WiFi-Positioning service. The vision service implements Canny Edge Detection, Hough Line Transformation and DBSCAN clustering in order to determine the vanishing point of the area flown in. When the vanishing point has been determined, it can be used to correct the yaw of the drone, allowing it to steer down a hallway, while keeping the nose of the drone correctly aligned. The distance service uses ultra sonic sensors mounted onto a 3D-printed hull with a Raspberry Pi Zero-W as a means of communication. This service controls the roll of the drone by using the distances measurements on either side of the drone to correct itself into the center. The WiFi-positioning service was implemented as a proof-of-concept of how the system could be used. In order to test the implementation of SharpFlying and the services, we designed a close-to worst-case scenario for an indoor drone. It had to navigate down a narrow hallway from different starting positions, where it had to centralize itself before landing in a designated area. The tests showed that the individual services can navigate the hallway, although with a lot of inconsistencies and mistakes. Combining the vision and distance service yielded better results and consistencies, where the drone only made one wrong landing.

Due to the services not allowing indoor navigation and the unpredictability of actions by the implemented services, we did not deem it safe enough nor ready to be used in a real scenario. Thus, in order to get an insight into Human-drone Interaction with an indoor autonomous drone, we used Wizard of Oz (WoZ).

With WoZ, we emulate similar behavior to that of a fully autonomous drone, however with the safety of an experimenter being able to immediately stop the drone in case of a dangerous situation. Thus, we designed an experiment with three different parts, each focusing on their own type of interaction. The first part is about primary interaction, where the participant is following an indoor navigation drone. The second part is an indoor area investigation, where the participant, using voice commands in a remote location, controls the drone between two locations, to then investigate an area and locate two hidden objects. The third and last part is about secondary interaction. In this task, the user is sharing a narrow hallway with the drone under different conditions.

Based on the interaction experiment, we withdraw four design insights, that future researchers within autonomous Human-Drone Interaction may consider when designing an autonomous system. This includes: *trust*, a metaphor explicitly stated by our participants. Trust deals with expected behavior of the drone. The participants expected the drone to be flawless in its navigation and ability to know the environment it is navigating. Furthermore, the participants wanted consistency in actions and speed of the drone. Lastly, we derive how much natural language understanding in necessary to properly interact with a drone using voice commands alone. Here we draw similar results to that of related work, that the interaction level needed is similar to communicating with a pet.

Currently, the implemented services of SharpFlying does not deal with any interaction specific measures, however, the implementation allows a service to be developed that does. Based on the results obtained through our study, the service would need to detect people and react to them by giving the person space and awaiting the person to move past the drone, before continuing. Furthermore, further research should be made into proper auto-stabilization in indoor environments, as the drone in the current state can be unstable, especially in intersections, however this could potentially be solved with a smaller drone.

We conclude that a multi-service framework for controlling autonomous drones can perform better than single instances of services. Furthermore, that allowing new services to be easily added to extend the autonomous system can provide better results. Based on the results of the exploratory study, several recommendations of valid services to extend SharpFlying has been proposed as design insights. For future work, we recommend continued development of SharpFlying to behave according to our design insights, to which the results of the exploratory study should be verified.

SharpFlying: A framework for autonomous indoor drone flight

Kasper Østergaard Helsted
Department of Computer Science
Aalborg University, Denmark
khelst13@student.aau.dk

Steffen Darby Carlsen
Department of Computer Science
Aalborg University, Denmark
sdca14@student.aau.dk

ABSTRACT

Drones are becoming increasing popular, however, the use of drones is limited to mostly outdoor usage. While research highlights new and smart technologies to perform indoor navigation, it is often focused on using a singular technology as a solve-everything solution. In this article, we designed and implemented SharpFlying, a generic and extendable multi-service framework for autonomous indoor drone flight. We implemented three proof of concept services based on related work to validate our framework implementation. We tested our services individually and together, which showed vast improvements when using multiple services at the same time. We contribute a framework for future researchers to use when developing autonomous drones with a multi-service structure in mind.

KEYWORDS

Drones, Human-Computer Interaction, Human-Drone Interaction, Indoor navigation, Autonomous flight, SharpFlying, Vision, Edge detection, Clustering

1 INTRODUCTION

The use of drones in indoor environment has been researched in regards to using a single technology to attempt and solve one of the many problems of indoor navigation. This includes: Recognizing tags using vision or RFID [8, 28], using deep learning for perception, guidance and navigation indoors [20] and a general use of vision to navigate the drone indoors [16, 17] or adding additional sensors to the drone [31, 33]. While some of the solutions has been developed as frameworks, none of them attempt to combine multiple approaches to indoor navigation into a framework for flying indoors with drones.

In outdoor environments, navigation is often done using GPS, gyroscopes, accelerometers and some wireless communication with a base station using the camera feed. These tools are great for outdoor navigation, but is not enough when it comes to indoor navigation, which can be used for in-building delivery, automated inventory and indoor emergencies [29]. Additionally, A survey made in 2001 shows that humans spend upwards of 87% of their time in indoor environments, this includes office buildings, airports, venues or homes. This result displays the need of indoor navigation and that any type of framework that allows indoor navigation has a practical use.

In indoor environments, a drone need to use its sensors and vision based systems to detect and avoid collisions. These are used in conjunction with obstacle avoidance strategies to find the best solution to avoid the environment [12].

Although a lot of research highlights indoor navigation strategies and how to approach this [13, 25, 28, 36], limited research could be found that designs and implements an indoor autonomous drone with the goal of detecting and avoiding humans [26]. However, their results does not focus on any interaction, but rather focus on the technical aspects of detecting and avoiding humans, with a singular contribution to Human-Drone Interaction (HDI), this being that the drone should move when meeting humans in a hallway.

In this paper, we have developed a standalone framework, SharpFlying. SharpFlying combines multiple results from previous work by combining vision, ultra sonic sensors and WiFi positioning into one framework that helps with autonomous indoor navigation.

The framework we have developed is not a fully implemented system, but rather a proof of concept. It is developed as a foundation to be build upon, where the system allows for easy expansion upon functionality by adding more services. This framework relies on the related work, and comes with a collection of developed libraries, in order to support the features which was deemed necessary for this framework.

The inspiration and some of the requirements of the project is based the SciRoc Episode 12 challenge - Fast delivery of emergency pills [3]. This challenge is about an aerial robot in an emergency situation, where a first-aid kit needs to be delivered to a customer in an indoor environment as fast as possible. The robot will need to automatically detect and avoid obstacles and account for GPS being unavailable. The challenge defines that the robot must navigate in an indoor environment, detect a customer and correctly maneuver in an environment with obstacles, land and deliver the item within a 2 meter accuracy and fly back. These requirements are the point of reference of requirements for the development of SharpFlying.

Initially, we will take a look at related work, to get an idea on how to approach the project and to get an insight into the current progress made by other researchers. Afterwards, we will go through multiple development cycles, each focusing on solving a problem within indoor navigation with a drone. Lastly, the different implemented services will be tested individually and in conjunction with each other, to show how each of them perform and compensate for each other.

2 RELATED WORK

In order to gain an understanding of current research in regards to navigation with drones, we look at the related work in regards to outdoor and indoor autonomous navigation with drones. This will give an insight into the current state of the art and novel approaches to solving outdoor and indoor navigation issues, with a focus on drones. The related work will be used to extract important considerations and guidelines for developing SharpFlying, our framework for indoor autonomous flight.

2.1 Outdoor autonomous navigation

Within outdoor navigation, there are several approaches that attempt to either solve or optimize the field of navigation. The conventional approaches use GPS to determine the position of the drone and either way-points or other planning algorithms to determine the most optimal route for the drone. Another approach that has evolved, especially in regards to race drones, is machine learning approaches using neural networks.

Conventional approaches

GPS is one of the most core technologies used for outdoor positioning and can be used in conjunction with a camera for fully autonomous flight without pilot input [24]. One of the most basic implementations of autonomous navigation includes way point planning, where the pilot determines a set of points the drone should navigate to, this feature is often implemented by the drone manufactures [4]. The approach of generating a flight plan based on points specified by the pilot has previously been researched in regards to precision [34, 35] and performance [19].

Carvalho et al. adapted a standard commercialized drone into an embedded system, without any reliance on a ground station for fully autonomous flight [11]. This was done using the Robot Operating System (ROS). Their work is a design and integration of the ROS framework with the robot's hardware using an Odroid micro computer mounted on the drone. Their results show the drone being able to correctly perform a simple mission consisting of performing a takeoff, flying in a square and then landing again using GPS only. The UAV responded satisfactory during random disturbances, such as wind and GPS signal fluctuation, with a minimal number of errors. They conclude that using filtering algorithms for GPS signal fluctuations, the minor inconsistencies in flight paths might perform better.

Machine learning & outdoor navigation

Some of the related work has started to look at machine learning as an approach for outdoor navigation. Most of their motivations stem from drone racing, where each drone is controlled by a human pilot, with a first-person view from the drone, who controls the drone at high speed using a radio transmitter. While human pilots need years of training to properly control the drone in fast paced environments, the skills are valuable to an autonomous system that must quickly and safely navigate through complex environments, which is the primary reasoning for using this as a task.

Simultaneous Localization and Mapping (SLAM) can provide consistent 2D to 3D pose estimation against a known environment, but is prone to errors during any accelerated movements, due to image blurring. SLAM is great in static environments where waypoints or movement trajectories are already defined or where speed is not a concern [10]. Kaufmann et al. describes Deep Drone Racing, a 2-part system for robust and agile flight of a drone in a dynamic environment [21]. Part one is a perception system, which uses a Convolutional Neural Network (CNN) to predict a goal direction in local image coordinates, together with a navigation speed. This is generated from a single image on a forward-facing camera. The second part is a control system. It uses the output from the perception system to generate a movement trajectory vector. Their results show that their system is able to navigate a complex race track, while avoiding common problems, such as drifting and a navigation in a dynamic environment.

Sadeghi et al. presented CAD$\hat{2}$ RL, a neural network trained purely on simulation data. Their goal is to train a neural network for obstacle detection, without having to put the drone into the real scenario [32]. Their results show that they can successfully fly and avoid obstacles. For future work they recommend including depth into the images using multiple cameras or combine their simulated training with real data, which could yield a better overall result.

Giusti et al. researched a drone as a medium of visual perception of a forest trail [18]. Here, they assume that as long the drone follow the trail, that it is free of any obstacles, thus focuses on finding and following the trail. They have trained a CNN on 17,000 training frames and uses that to process an image in order to obtain a perception of the trail. They derived a two-class problem, one has to decide whether the trail is visible or not, while the other class determines where the trail is. The biggest problems they faced were in regards to image quality, where their training data was recorded from GoPros mounted on a researchers head during a regular hike. This meant that the fast-paced motion pictures from the drone caused complications with the classifications. Furthermore, differences in light strength across the trail caused the drone to be unaware of its surroundings, which caused some crashes. In general, they were able to classify trails correctly in synthetic tests and that their system performs better than any alternatives and to a certain degree, comparable to humans.

2.2 Indoor autonomous navigation

There is a fairly recent body of work attempting to solve the problem of indoor navigation in GPS denied environments using one of the Parrot drone platforms [9, 16, 22]. The solutions proposed includes: the use of markers for vision to detect, vision based distance estimation and full on image processing systems.

In recent years, the use of drones has expanded into real-life applications, especially in outdoor scenarios, such as monitoring and surveillance [1, 6]. Thus, research has been made in regards to HDI in outdoor environment with respect to: how users interact with drones and how they expect the drone to respond to their interaction. However, only limited research has gone into the potential interaction in indoor environments. Lioulemes et al. researched the safety challenges when humans and drones collaborate in indoor environments [26]. Their results show that humans expect the drone to behave in a similar manner to a ground vehicle, that is, that the drone should be the one maneuvering away from the human when possible.

Follow-me drone

Mao et al. investigated the possibilities of an indoor follow-me drone [27]. Their motivation is that video taping is a costly and time consuming process with possibilities for automation. They describe the challenges of being unable to use GPS or similar technologies to properly do indoor tracking. Their solution uses robust acoustic tracking to determine the distance between the user and the drone. Their results show that it is possible to find the location of a user and follow them within a specific distance, with the limitations

being unable to properly follow the user during extended periods of disconnections with the sound signals, I.E. hard concrete or a human blocking the signal [27].

Machine learning & indoor navigation
Duggal et al. researched the use machine learning in contrast to fully autonomous indoor navigation for drones [13]. They use a single monocular camera on the front of the drone. They input an image into a Hierarchical Structured Learning (HSL) algorithm that outputs a depth map. This depth map is fed into a CNN to generate the flight planning commands for the drone. Their results show that they use ≈ 180ms to perform HSL and ≈ 200ms for the CNN to generate flight planning commands. They specify that the Bebop drone they use for their experiment takes approximately 400ms to respond, so that their computation time is justifiable. Their flying experiments show that they have an 82% successful navigation rate, however that they had problems with the drone drifting in the indoor environment and problem with depth map estimation due to changing light levels in the hallways.

Vision-based approaches
Adriano Garcia & Kanad Ghos researched autonomous indoor navigation using a stock quadcopter using off-board control [16]. Their approach to indoor navigation uses vision to determine the location of floors and walls. This is done using Canny edge detection in correlation with Hough Line Transformation. This is done to form vanishing points. The vanishing points provides a general central marker of the image and can be used for yaw correction and the position of the vanishing lines can be used as a reference for roll correction. One part of their results show that they can detect an intersection within a 50cm difference of their measured estimate. The second part of their results is in regards to navigating a narrow hallway. In total they had 20 test flights, with 15 total collisions, with only 2 of them being hard crashes. They identify the error causing the hard crashes in both cases, saying that the implementation of 180 degree turns will fix this. They manage to successfully detect, stop and turn at the intersection in 90% their tests. Because of their implementation heavily relying on being able to find vanishing points for navigation and stabilization of the drone, during turns, they had troubles correctly finding the new vanishing point, causing problems in 25% of their test cases. A similar type of research was made by Bills et al. who focused on indoor navigation in narrow staircases [8]. Their results show that their vision based solution succeeded over 80% of their tests, however that they had problems getting the drone to properly fly in the indoor environment causing air drifts and turbulence, which caused the drone to crash or touch the walls.

Distance sensor-based approaches
There already exists different types of solutions that attempt to support drone flight using distance sensors. This includes a obstacle detection and collision avoidance system using ultrasonic sensors by Gageik et al. [15]. They show that ultra sonic sensors perform well when reflecting onto straight surfaces, where it is capable of reproducing the environment with an accuracy of centimeters, however that they are unable to reliably detect objects beyond 250cm. They recommend using additional sensors, such as infrared sensors, to account for some of the negative aspects of using ultra sonic sensors. The use of infrared sensors as an obstacle avoidance tool has previously been researched [14, 23]. Their results show the use of the range finder in conjunction with the inertial measurement unit, that they can stabilize the drone in an indoor environment within a 2x3 meter area. They conclude that this is adequate, since the rooms they navigate in are always at least 6x6 meters. In one of the test cases in [14], the drone is flying in a narrow hallway towards a dead end. In this test, their system showed inconsistencies due to wind turbulence in the narrow indoor environment. Even though they were able to correctly navigate the area and avoid obstacles, specifically tweaking the system variables was needed in order to accommodate the narrow areas.

3 SHARPFLYING

Based on the related work, we have chosen to focus on three technologies. The first technology is vision, this is chosen based on the promising results of Adriano Garcia & Kanad Ghos, who managed to perform indoor navigation using deterministic well-researched vision models. The second technology is distance using ultra sonic sensors. This is implemented in order to accommodate for the weaknesses of vision. These sensors measure the distances between the drone and environmental objects, allowing us to stop the drone from crashing into walls. Furthermore, Bills et al. described scenarios where they had problems with the air generation of the drone when indoor. We attempted to replicate their results by manually flying indoors, which showed that when the environment changes, I.E. flying past an intersection, the drone can no longer stabilize itself during hovering, causing it to increase its speed. Using ultra sonic sensors, we can detect drastic changes in the environment and accommodate for these environmental changes. Lastly, some of the distance sensor-based related work showed promising results in regards to indoor environments and that their systems are able to function in even narrow environments, as long as the system parameters are designed to accommodate for this. The outdoor approaches use GPS in order to perform self-localization of the drone, this is not available for indoor flight. The related work simplify their experiments by ignoring the need to navigate a building between two known points, but rather follow a user or move inside a simple environment, such as a hallway with no obstacles. In order to implement indoor *navigation*, we will use WiFi positioning as the third technology. The goal is not to get an absolute positioning of the drone, but rather self-localization to the point of the drone understanding where it is, such that it knows how to move between two known locations.

4 HARDWARE

This section describes the drone and the base-station used for the system. The drone chosen for this project is a Parrot Bebop 2 (42cm x 39cm x 9cm) modified with a custom 3D printed hull. Furthermore, the drone has 3 ultra sonic sensors (HC-SR04) mounted onto the hull. The ultra sonic sensors measures in a 15°angle and has a range of 2cm to 400cm.

The drone with all modifications can be seen in Figure 1. The ultra sonic sensors are connected to a Raspberry Pi Zero W, which

is also mounted on the drone. The Raspberry Pi is running Raspbian as its operating system.

Figure 1: Image visualizing the drone setup, with ultra sonic mount.

The Raspberry Pi offloads all of the heavy computations to a base-station. The base-station is a laptop with an i5-7300HQ @ 2.5GHz processor running Windows 10. During integration tests of the distance sensors, it was discovered that the drone propellers emit ultra sonic sound, causing some of the sensors to give invalid data. Due to this being discovered late in the project, the tests were performed by taking the hull off the drone, placing it on an office chair and imitating the movements of the drone.

5 IMPLEMENTATION

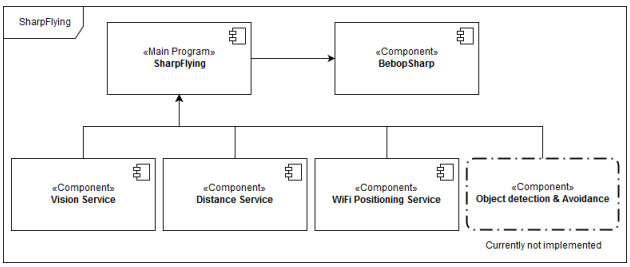

Figure 2: A simplified overview of the SharpFlying component architecture

The implementation consists of 4 different modules. These are the modules outlined by the related work as necessary to implement an indoor autonomous drone. The system architecture can be seen in Figure 2.

SharpFlying is the entry point for the system. Here the declarations of the different services are made and the results from each service is retrieved. BebopSharp is the library developed to control the drone. The Vision service contains the implementation of Canny Edge detection, Hough Lines and the DBSCAN clustering. The Distance service contains the implementation of the ultra sonic sensors. The WiFi Positioning service provides functionality to find an estimated position of the drone inside a building. All of the services return a `Response` together with a confidence value. A confidence value is a value that indicates the services confidence in its result. From the vision service, the confidence value is dependant on the changes between images. If we observe a large change in the results from a captured image, the confidence value will drastically go down. This will handle cases of blurred images or other cases of bad data.

The entire point of SharpFlying is to be a fully asynchronous multi-service framework. This means that all services are running separately and "as fast as possible", causing no services to bottleneck each other. This means that there is no guarantee of a looped order of actions, but that one service can publish multiple results before another service publishes theirs.

5.1 BebopSharp

BebopSharp is the implementation of the drone controller library. This is the library implementing all of the functionality we can perform with the drone. Initially, we implemented the bebop system in Python using PyParrot, an existing library for developing drone flight applications [7]. However, during integration of the vision service, the performance of Python failed to deliver the expected results, with this implementation only being able to process 1-10 frames per second, depending on the amount of data in each image. Thus, we decided to rewrite the entire project in a compiled language, namely C#, due the project members having previous experience in the language. However, there existed no library equivalent of PyParrot for C#. Thus we created our own library to communicate with the Bebop drone, called BebopSharp. After successfully connecting to the drone, a thread that generates the drone command packets is started. This thread ensures that if the drone retrieves a `FlightVector` (An object containing movement values) from SharpFlying, that the drone performs the corresponding movements. The amount of times the command generator generates and sends commands is controlled by the Constructor call to the BebopSharp class, where the user can specify the number of updates per second, per default this is 10.

In order to ensure safety, a thread watcher is used. The thread watcher ensures the command generator thread is active and performs an emergency landing if we lose connection to the drone.

The drone communicates using UDP between itself and the client. Based on the ARSDK Protocol documentation (Official documentation of how the drone works internally), the communication with the drone has been implemented to handle pings from the drone (the drone asking if we are still alive) and when the drone has updates to its sensors [30]. This is implemented by recursively withdrawing the first 7 bytes, containing the packet header, then based on the total packet size, we store the bytes from the data

	Services		
	Vision	Distance	Indoor positioning
Objective	Detect features in rooms	Measure distance to remote objects	Estimates location from WiFi strength
Implementation	Uses multiple vision based algorithms to locate room features.	Uses an external mounted system to measure distance. Using multiple measurements, to understand distance.	Uses external mounted system scan for WiFi and comparing to know hotspots, for positioning.

Table 1: Simplification of implementation of different services

array from the drone. This is performed recursively since the drone can send multiple data packets at a time.

In order to parse the drone-UDP-packet, we start by parsing the `Datatype` byte of the packet. The `Datatype` tells us how to respond to the packet retrieved, this is either: Ping, acknowledgement packet (The drone asking us whether we retrieved a specific packet) or sensor data. The next byte in the header is the `Buffer ID`. This specifies if the data is an internal drone command, a data buffer or an acknowledgement buffer. Afterwards, we have the `Sequence ID`. This is an identifier for the packet. Lastly, we have the `Packet size`. This defines is the total size of the entire data structure as a little endian integer.

In order to send a `FlightVector` to the drone, we must first determine how the drone should move. In order to ensure the horizontal movement (Yaw & Roll) of the drone, the camera from the drone is used to extract relevant data from the images. The implementation of this is described in subsection 5.2. Vision does not provide any details about distance, thus in order to ensure the drone is in the middle of the corridor, we use ultra sonic sensors in our distance service. The implementation of this can be seen in subsection 5.3. In order to navigate indoors, we have to know where the drone is located, this is done using indoor WiFi positioning, the implementation can be seen in subsection 5.4.

5.2 Vision service

Vision allows us to extract information about the environment that distance-based sensors or other measurement devices can not directly provide. The vision service will be used to extract details from the live camera-feed of the drone.

The vision service extends the base-service, overriding the `Input`, `Run` and `GetLatestResponse` methods. These are used to give data to the service, start the service and retrieve the latest result from the service.

The Vision service implements Canny Edge Detection and Hough Lines Transformation. In order to visualize the data, `RenderGeometryLib` is used. This library is a wrapper around the OpenCV drawing functionality and allows us to draw geometric figures on a frame and render it. `DBSCANLib` is the clustering library used. We are using DBSCAN clustering algorithm over the traditional K-means, since we can not estimate or predict the expected amount clusters in the image. `GeometryLib` contains base implementations of `Box`, `Line`, `Point` and `Polygon`, which is used throughout the implementation of all features. An example can be seen in `EdgyLib`, where the implementation of Hough Line Transformation uses `Line` to represent its result.

Implementation

Based on the previous research described in the related work, we decided to build upon the approach described by Adriano Garcia & Kanad Ghos in [16]. This means that we take the raw image from the drone, downscale it to 640x360px, apply Gaussian blur and run Canny Edge detection on the image.

Canny Edge detection consists of 5 different steps. The first step is to apply Gaussian filtering to smooth out the image, this removes the noise. The next step is to determine the location of the horizontal, vertical and diagonal edges in the blurred image. Afterwards, a scan of the image is performed to remove unwanted pixels not apart of the edge. Furthermore, each pixel is checked if its a local maximum or if there is a local maximum close-by the gradient. The result of this step is a binary image with thin edges. The last step decides which of the edges are useful. For this, two values, maximum and minimum threshold are used. The edges that lie between the threshold values are decided to be edges based on their connectivity.

After Canny Edge detection, we find the lines in the image using HLT. HLT uses the binary representation of the image from Canny Edge detection and based on: a minimum distance between points, a maximum angle between the points in a line and a minimum number of points to represent a line, the lines are returned.

After finding the lines in an image, we will need to find the point with the most intersections. If there exists intersections in the image, we perform DBSCAN clustering based on the intersections. This gives us a best-case centroid point, which is our vanishing point. If the vanishing point is outside of the center-location of the drone's image, the drone will auto-correct itself to have the point in the middle. An example of the steps of the vision service can be seen in Figure 3.

The top-left image is the raw input image, taken directly from the video feed of the drone. The top-right image is the result of performing Canny Edge detection on the image. The bottom-left image is the result of HLT, where we find the lines in the image. The bottom-right image shows the clustering, where the dots indicate points and the red dot is the centroid.

5.3 Distance measurement service

Using the Vision service, the drone has an understanding of the environment and can use this to auto-correct itself. However, in more advanced environments, the drone requires understanding of distances to accommodate for changes in the environment, that vision can not detect and classify. The distance service uses 3 ultra sonic sensors, one on the front and one on either side of the drone. The drone and its extra components is described in section 4.

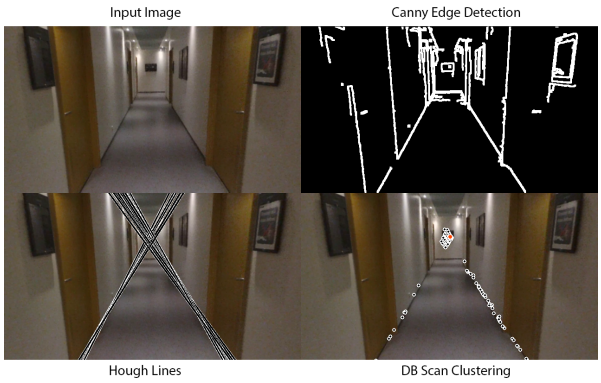

Figure 3: The four different stages of which our Computer Vision sees each processed video frame

The distance service extends the base-service, overriding the `Run` and `GetLatestResponse` methods. These are used to start the service and retrieve the latest output from the service. It is dependant on `UDPBase`, which is a wrapper around the C# UDPClient, to which it implements a network handshake and handles disconnects. This is implemented for easier communication over UDP across multiple classes and to reduce code duplication across services. The Raspberry Pi sends the data to the base-station, which uses the data to compute an action for the drone.

Implementation

The implementation of the distance measurement service uses a Client-Server architecture. The server runs on the Raspberry Pi, and the client is a service for the SharpFlying system.

In order for us to use the ultrasonic sensors, we have mounted a Raspberry Pi Zero-W on the drone, which uses General-purpose input/output (GPIO) to get the distance from each of the three ultrasonic sensors. The Raspberry Pi is also equipped with an UDP server, which is used for transmitting the data to all clients, which are listening. When a client connects to the UDP server, a handshake is made to register the client as a subscriber to the sensor data.

The UDP server reads from the sensors 10 times a second. Since ultrasonic measurements can vary depending on the surroundings, we average the distance measured over a period of 10 measurements, so highly varying results does not have a large impact, but if an object suddenly comes close to the sensor, the sensor will still give enough small values for it to have an effect on the result.

The server transmits the latest result four times a second to all clients as JSON.

The client subscribes to the UDP server running on the Raspberry Pi and receives JSON data as fast as possible, which is then mapped to internal objects. The client calculates the correct action for the drone to perform, based on the data received from the server.

The actions are separated into two different types of movements; critical and non-critical movements. The critical movements are executed first. This is done by checking if any of the sensors detect objects within a "safe distance" of the drone. The default safe distance is 30cm to either side. If an object is detected, the drone should move away from that object immediately. If no critical movements are needed, the non-critical movements are calculated. An example of a non-critical movement is to center the drone in a given area by using the ultrasonic sensors on each side of the drone. All of these movements are calculated in the distance service and is returned to the SharpFlying program as a `Response`, where the movement vector is given with a confidence value. The confidence value by the distance service is hard-coded to 75 out of 100. This means that we are 75% sure that the value retrieved from the service is *correct*. Since the confidence value of the vision service is fluctuating, this allows it to both be below and above the confidence of the distance service, making it possible to be at a bigger priority than the result from the distance service. The confidence value does not cause a result to be ignored. Instead, a higher confidence value causes the final movement vector to be more impacted by the service result.

5.4 Indoor Positioning Service

At the current stage of the implementation, we have made the drone become aware of its surroundings, however, it has no understanding of its position in the building. Thus, in order to navigate a building and move between two locations, the drone must be able to position itself. Following the specifications of the SciRoc challenge, we should assume that the area we are flying in is GPS restricted, thus another approach is used.

Through previous indoor navigation research made at the House of Computer Science, Cassiopeia, we obtained an IFC (Industry Foundation Classes) File, containing router locations and their corresponding MAC addresses.

Implementation

The implementation of the indoor positioning service uses a Client-Server architecture, similar to that of the distance service. The server is the Raspberry Pi Zero-W, which was previously described in section 4. The client is a service written for the SharpFlying system.

On the **server**, the operating system has two commands, `iwlist` and `egrep`, which allows for the scanning of nearby WiFi access points, and filtering of command output. These commands are run four times a second, to get the most accurate result. The output of the commands is parsed and mapped to an `AccessPoint` object.

The `AccessPoint` object contains information about the `ESSID`, `Signal Strength`, `Quality`, `Mac Address` and `Frequency` of the access point.

This data from the `AccessPoint` object is serialized into JSON and transmitted to all clients using the same UDP server as the distance measurement service.

The **client** subscribes to the UDP server and receives a list of `AccessPoint` objects serialized as JSON as fast as possible. When the client parses the data from the the server, the union of the retrieved access points from the service and a list of known access points is found. The list of known access points has an associated latitude and longitude, which is used for calculating the distance from the drone to each access point.

After the distances have been calculated, the access point with the smallest distance is then selected as the assumed area, where the drone is within. We have chosen to not use trilateration, due

to intermediate test results showing that due to interference from surroundings, the second nearest access points diameter was greater then the smallest diameter and offset, thus no intersections was found. In order to counteract this and due to only using the result as a general approximation of distance, the access point with the smallest distance was selected as the position of the drone.

6 EXPERIMENT

As presented in section 5, SharpFlying contains multiple services. Each of these services provide their own functionality to the overall system and work independently from the other services. The experiment will test the vision and distance services independently. This tests the two service's ability to perform its intended action, for example, the vision service to modify the yaw of the drone. Furthermore, the services' will be tested in conjunction with each other, to show how the services compensate each others weaknesses.

We chose a test case that tests the system's ability to perform fully autonomous movements in an indoor environment. Each service and both services combined will be put through the same test case. This will show the strengths and weaknesses of the services.

Test cases

In total, the system is tested in three similar test cases. The drone is placed in varying position in a hallway. The positions are: Center of the hallway and 25% of the total distance towards both right and left side of the hallway. The change in placement is made to test the properties of the system to place itself in the center of the hallway using the different services.

6.1 Environment

The environment is a hallway (156 cm x 800 cm) with solid walls on either side. In the hallway, there is an window at the end of the test area with direct sunlight during the tests. A figure of the test area can be seen in Figure 4.

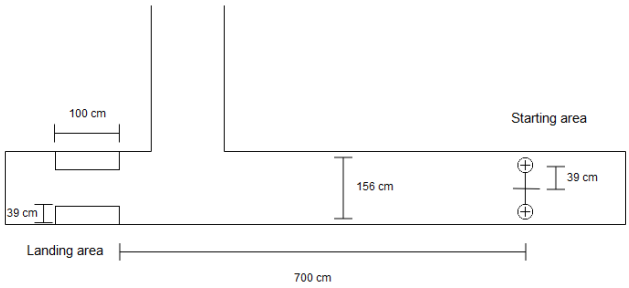

Figure 4: The environment used for the autonomous flight tests

In the starting area, three positions are marked, the center position and the two locations to the right and left side of center. The left and right positions are placed 25% of the total distance from the center starting point equalling to 39 cm. The landing area is a 100 cm by 78 cm area. Since the goal of the drone is to automatically center itself, the same deviation used during take-off is used for the landing area. The length of the landing area, 100 cm, was chosen due to the landing being performed manually by the experimenter from a distance, giving some room for human error.

6.2 Data analysis

In the experiment, there is one independent variable, the service(s). The dependant variables that will be used to determine the effect on the service(s) are:

- Task completion time
- Number of errors
- Successful landing

Task completion time is measured from when the drone take-off command is send to when the drone is landed. The time will be measured using the `Stopwatch` class in C#. The number of errors is incremented every time the drone: Hits a wall or otherwise has any collisions with objects in the hallway. If the experiment ended in a full crash, it will be noted down in the table of results. The last dependant variable is whether the landing was successful or not. A successful landing implies the drone being at the end of the hallway and landing with a maximum 25% offset from the center. All of the data was collected from video recording of the experiments, output logs from SharpFlying and observers notes.

The data is analysed by initially extracting the main dependant variables from the video material. Afterwards, each test-run is analysed to extract relevant flight information. The relevant flight information relates to the cause of the drone's actions. This allows us to create a timeline of input to the services and actual output versus expected output. This will give an insight into why the drone made any given decisions.

7 RESULTS

In order to analyze and interpret the results from our tests, video were recorded of the different flights and the controller screen.

In Table 2, a summery of the results from our tests can be seen. From this table, it can be seen that the completion time of the tests were all within a small margin of error, however, that both the vision and distance test varied the most in regards to time.

	Services		
	Vision (n=9)	Distance (n=9)	Vision + Distance (n=9)
Task completion time (seconds)	10.74 (2.25)*	9.7 (1.44)	10.79 (0.85)
Number of errors	5	2	0
Successful landings	2*	5	8

Table 2: Combined results from test flights, * n=6

The number of vision tests, in regards to completion times, is lower then the initial 9 tests, which were performed. This is due to the drone crashing beyond recovery, which would immediately stop the current test run. Likewise, when looking at the errors and number of successful landings in Table 2, the distance and vision service performed the worst, with substantially worse results than *vision + distance*. Based on the results, three different themes are

withdrawn, these are: Service validation, lack of distance awareness and correction with multiple services.

Service validation
The first theme shows that we are able to successfully validate the data input and output of the distance service. In order to validate the precision of the distance sensors, unit tests were performed. Two different sensor validation tests was made, distance reliability and movement vector test.

The distance reliability test shows the precision of the sensor and the reliability of the service. The movement vector test, is a test where the distance service outputs the movement vector, and this vector is compared to an expected vector, depending on the drone's actual distance. This movement vector is corresponding to the movement the drone should perform. These tests show whether the distance service is calculating and outputting a correct movement vector.

The distance reliability test is done by placing the drone precisely 100 cm from a wall. The distance service then measures the distance 100 times and then output the minimum, maximum and average values. The results can be seen in Table 3. The results show that

Distance unit test	Distance (cm)
Minimum distance	99
Maximum distance	102
Average distance	100

Table 3: Unit tests of the distance service

the implementation of the service works as intended and provides stable results in a calm environment.

The movement vector test is done by placing the drone in a hallway. Then the distances on either side of the drone is measured with a ruler and observed using the ultra sonic sensors. The output vector from the distance service is then outputted and compared to the expected vector. This movement vector is verified to be equal to the expected output. An expected value is a movement vector with a direction towards the side with the greater distance.

Lack of distance awareness
The second theme shows the general inability to understand distance and overall interpret hallways using the individual services alone.

This could be seen in the distance service, where the drone would lose track of the direction, depending on the starting position. This would in some cases make the drone rotate uncontrollably, and unable to recover to the initial direction.

In total we experienced 4 successful and 5 unsuccessful flights using the distance service. During landing, in 7 of the tests, the drone was aligned with the hallway, but in 2 of the flights, the nose of the drone would not align with the hallway upon landing. These results show, that the distance service by it self was not able to detect the direction of the hallway in which it was flying. This could be observed since the drone would deviate in the yaw direction, and not recover from it. The problem most commonly occurred when the takeoff was off center, which causes the drone to try and re-center itself. This causes over-corrections and made the nose of the drone be unaligned with the hallway.

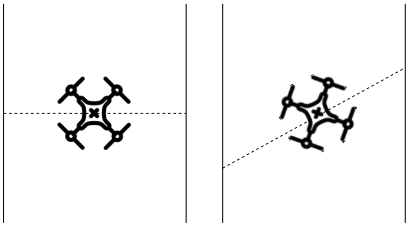

Figure 5: Illustration of distance service in hallway

In Figure 5, the hallway feature problem can be seen. This figure shows how the distance service is unable to understand if the hallway is getting wider or the drone is rotating in the yaw direction.

The results form our vision service shows that the drone would correctly change its yaw to center itself, but would drift, causing major inconsistencies in its movement direction. During the tests, the drone crashed to an unrecoverable state once, and were operator aborted twice. The crash happened while the drone attempted to correct its yaw, while drifting directly towards a door frame. This caused the drone to hit the door frame and crash the drone. In the 2 aborted tests, the operator deemed the drone was moving towards an unrecoverable state. In the first aborted test, the drone was moving into an intersection, where an ongoing movement would cause a direct frontal crash with a wall. The errors observed during the tests were caused by the drone zig-zagging between the walls towards the end of the hallway. These inconsistencies translated into the drone missing the landing zone. In general, the drone using only vision is unable to properly center itself in the hallway and will instead drift, which means the drone's movements are somewhat randomly determined.

Correction with multiple services
The third theme shows that using multiple services, the overall results drastically improve. It was clear that using a single service, would not be sufficient for autonomous flight, since each service in itself have different weaknesses.

For example, the vision service is unable to account for the distances in the environment, thus colliding with the walls. It does, however, properly find the center point of the hallway and can center itself. The distance service was able to traverse the hallway with no hard crashes, but with the nose of the drone misaligned from the center, causing the drone to slide sideways and missing the landing zone. In the *vision + distance* test, the two services run at the same time, in order to compensate for each others limitations.

The result from the *vision + distance* test showed that in total, 8 out of 9 tests are successfully completed and 1 test where the landing zone was missed.

The *vision + distance* tests showed the drone is able to navigate a hallway using `SharpFlying` and its developed services with only one unsuccessful landing and no errors during the flight. In the test, where it failed the landing, the drone compensated too much from the air generated during lift-off from the right side, causing it to steer too far left. This caused the vision service to temporary lose

the center location of the hallway, which meant that the natural drift left got the drone close to the wall, causing an emergency roll towards right. At the same time the yaw was right-heavy, this meant that the drone barely managed to avoid a wall-collision, but due to the limited length of the hallway, it did not have enough time to get to the center of the hallway before the landing zone.

	Vision (n=6)	Distance (n=6)	Vision + Distance (n=6)
Task completion time (seconds)	10.81 (2.75)*	9.11 (1.34)	10.54 (0.97)
Number of errors	5	2	0
Successful landings	1*	1	5

Table 4: Table showing results from tests, where the drone started off-center, * n=4

As can be seen in Table 2 and Table 4, when the drone started in a non-center position, it had a higher degree of non-completeness and a higher margin of error. The time differences are similar between all starting positions. However, when looking at the margin of error in Table 2, the error rate is similar to Table 4, however the sample size is 33% smaller, which makes the errors vastly more influential. Also the amount of successful landings suffers from the same problem as the errors. The drone is vastly better at correcting the position when using multiple services.

Limitations

During our tests some limitations became clear. The vision service is limited to travelling through a narrow hallway, where it is able to successfully identify the vanishing point and use that to auto-correct itself for successful navigation. The system has no understanding of distances. This means that it is unable to determine when it is too close to the sides and end of a hallway.

The distance service is unable to understand the features of the hallway. This causes the drone to rotate uncontrollably, making it unable to recover the nose to the initial direction.

We experienced turbulence when approaching the intersection in the hallway due to the air generation of the drone. Using only the vision service, the intersection is not being detected, which means we can not compensate for this.

8 DISCUSSION

Through the related work, we found that research has been made in regards to indoor navigation and indoor flight using a multitude of different approaches. However, none of the work used multiple different approaches in a singular framework. Thus, we designed and implemented SharpFlying, a generic and extendable framework for indoor flight. Our results show that each of our implemented services separately show strengths and weaknesses and that by combining multiple of these approaches, that the amount of weaknesses can be drastically limited. The results can be derived into two main themes: Standalone services lack awareness and accuracy increases with multiple services. Lastly, we will discuss the implementation in regards to optimizations related to the implementation itself and the vision service.

Standalone Services lack awareness

Our test results show, that vision service alone is able to navigate the hallway, while only modifying the yaw of the drone from all starting positions. However, it is unable to properly correct itself to be in the middle of the hallway and thus misses the landing zone 7 out of 9 times. Furthermore, if the drone experiences turbulence at the lift-off moment, the drone has a big chance of colliding with the walls or completely crashing. Similar test results was found by [16]. Their results showed that their vision system could lose the vanishing point which caused collisions. Our tests showed that having no understanding of distance or the depth of the room, caused the drone to have major inconsistencies in regards to staying centralized in the hallway. The test results show that the drone is able to find the center point of the hallway and point the nose of the drone in that direction, by modifying the yaw of the drone. However, it can not properly account for turbulence or drastic changes in terms of distance to the walls. We believe that vision based systems, which are stand-alone, should explore the use of machine learning approaches more-so than line detection. A standalone CNN that generates flight commands based on a depth map was created by [13], their results showed a successful navigation rate of 82%. However, they had issues with drifting, similar to how our vision service experienced.

The distance service results showed that the distance service alone makes the drone able to move between two locations by only modifying the roll of the drone. The drone had three fully completed flights, where the drone was able to correctly stay center of the hallway. Coincidental, the successfully completed tests were all when the drone started at the center position. This shows that when the drone is already center, that the distance service is good at making the drone stay there, but when the drone is starting uncentered, that it fails to either successfully land, or has collisions with the environment. The test cases, where the drone did not start in the center, showed that the nose of the drone would be misaligned with the center of the hallway, causing the distance service to overcompensate the roll values, which meant the drone was unable to perform a successful landing, even though it was able to reach the landing zone. In general, these results were expected from the ultra sonic sensors, especially considering the related work of [14], where they had to fine-tune their distance sensor algorithm and still had problems in narrow areas due to the air generation of the drone.

Accuracy

The test results of using both vision and the distance service show that by combining multiple services and using both of their results to navigate the hallway, that the amount of errors are minimized. This causes the drone to successfully recover from almost-impact situations[1]. The video linked in the footnote shows the drone recovering from a non-center starting position using the both implemented services. This test results show that the drone was able to successfully navigate the hallway and avoid collisions.

[1] https://streamable.com/a0igt

When testing both services together, there were 0 errors and 1 missed landing. This missed landing was because of the drone compensating too much from the air generated during lift-off from the right side, causing it to steer too far left, making it unable to recover before landing.

Optimizations

One of the main goals of the implementation was to achieve near real-time data processing. Thus, we had to optimize the implementation into a state of "as fast as possible". An example is with the vision service: While an image is being processed, any *new* image will be added as the newest available image and the old image will be discarded. This causes us to only ever use the newest available data and not have the system halt behind. Since we always assume that there is a context between the latest processed frame, whatever frames we discard should not cause the system to behave overall differently. The queue system, combined with dynamically modifying the input parameters to the primary bottleneck of the system, HLT, caused the vision service to achieve a processing frame rate equal or greater than the camera feed of the drone (30 FPS).

Currently, the services, we have implemented only uses the CPU for image processing. However, it is possible to offload some of the computations to a GPU using for example CUDA with OpenCV. According to the OpenCV documentation, most primitive image processing techniques can see speed increases of up-to 30x using CUDA [5]. However, using CUDA comes with a computational cost of moving the image from main memory to the GPU's memory [2].

In this study, we designed and implemented a framework for autonomous indoor flight with a drone. The framework is designed to be generic and extendable with further services. Testing of the implemented services showed that they individually are able to perform some part of the necessary actions to perform indoor flight, but that their weaknesses far outweigh what they can do individually. In the combined service tests, where the results of both services are used, the drone shows drastic better results and is able to successfully navigate the indoor hallway and recover from almost collisions in a narrow hallway.

9 CONCLUSION

Considering the increase of research in regards to indoor drones and that most of the current studies focus on finding a singular, solve everything solution, for indoor navigation, we took some of the related work's ideas and implemented them into a generic and extendable framework, SharpFlying. The framework was unit tested, to discover the strength and weaknesses of each service, which showed that each service was able to account for its own part of the control, but that the drone was unable to properly navigate a hallway consistently. We found that combining the results of our two services, vision and distance service, the framework was able to successfully navigate the indoor hallway with only a single missed landing and 0 errors during flight. We contribute a framework that future researchers can use to implement their autonomous indoor navigation systems, allowing them to easily create multi-service structures for controlling their drones.

Currently, the system assumes that the developer accounts for the results from a service themselves and knows how to convert the results of a service into a valid movement vector. The two services we implemented each control a direction, but there are no overlap between them. Given further implementation, it would be interesting to automatically choose the correct movement based on a level of confidence from the service, in the case that multiple services would attempt to change a singular movement direction, for example roll. Furthermore, we would have liked to finish the implementation of a third service into our service structure, namely WiFi positioning. This would have allowed us to navigate indoor, rather than just fly and account for changes in the environment. We researched the use of WiFi positioning and managed to retrieve an approximate position in the building, but did not have enough time to combine the results of the positioning and navigation.

REFERENCES

[1] [n. d.]. Aerial Surveillance and Monitoring, observation with UAV / drone. https://www.microdrones.com/en/industry-experts/security/monitoring/

[2] [n. d.]. CUDA Kernel Overhead. https://www.cs.virginia.edu/~mwb7w/cuda_support/kernel_overhead.html

[3] [n. d.]. E12 - Fast delivery of emergency pills | SciRoc. https://sciroc.eu/e12-fast-delivery-of-emergency-pills/

[4] [n. d.]. Flight Plan | Parrot Store Official. https://www.parrot.com/us/flight-plan#customized-flight-plans

[5] [n. d.]. OpenCV CUDA documentation. https://opencv.org/cuda/

[6] [n. d.]. Surveillance Drones | Electronic Frontier Foundation. https://www.eff.org/issues/surveillance-drones

[7] Amy McGovern. [n. d.]. PyParrot. https://github.com/amymcgovern/pyparrot

[8] Cooper Bills, Joyce Chen, and Ashutosh Saxena. 2011. Autonomous MAV flight in indoor environments using single image perspective cues. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 5776–5783. https://doi.org/10.1109/ICRA.2011.5980136

[9] Pierre-Jean Bristeau, Francois Callou, David Vissière, and Nicolas Petit. 2011. The Navigation and Control technology inside the AR.Drone micro UAV. *IFAC Proceedings Volumes* 44, 1 (1 2011), 1477–1484. https://doi.org/10.3182/20110828-6-IT-1002.02327

[10] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. 2016. Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. (6 2016). https://doi.org/10.1109/TRO.2016.2624754

[11] Joao Pedro Carvalho, Marco Aurelio Jucá, Alexandre Menezes, Leonardo Rocha Olivi, Andre Luis Marques Marcato, and Alexandre Bessa dos Santos. 2017. Autonomous UAV Outdoor Flight Controlled by an Embedded System Using Odroid and ROS. Springer, Cham, 423–437. https://doi.org/10.1007/978-3-319-43671-5{_}36

[12] Boris Crnokic, Snjezana Rezic, and Slaven Pehar. 2017. Comparision of Edge Detection Methods for Obstacles Detection in a Mobile Robot Environment. https://doi.org/10.2507/27th.daaam.proceedings.035

[13] Vishakh Duggal, Kumar Bipin, Utsav Shah, and K. Madhava Krishna. 2016. Hierarchical structured learning for indoor autonomous navigation of Quadcopter. In *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing - ICVGIP '16*. ACM Press, New York, New York, USA, 1–8. https://doi.org/10.1145/3009977.3009990

[14] Nils Gageik, Paul Benz, and Sergio Montenegro. 2015. Obstacle Detection and Collision Avoidance for a UAV With Complementary Low-Cost Sensors. *IEEE Access* 3 (2015), 599–609. https://doi.org/10.1109/ACCESS.2015.2432455

[15] Nils Gageik, Thilo Müller, and Sergio Nieto Montenegro. 2012. Obstacle Detection and Collision Avoidance Using Ultrasonic Distance Sensors for an Autonomous Quadrocopter. https://www.semanticscholar.org/paper/Obstacle-Detection-and-Collision-Avoidance-Using-an-Gageik-M%C3%BCller/65f460229474e64687ecb209f6965d45bd7d5c34

[16] Adriano Garcia and Kanad Ghose. 2017. Autonomous indoor navigation of a stock quadcopter with off-board control. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE, 132–137. https://doi.org/10.1109/RED-UAS.2017.8101656

[17] Adriano Garcia, Edward Mattison, and Kanad Ghose. 2015. High-speed vision-based autonomous indoor navigation of a quadcopter. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 338–347. https://doi.org/10.1109/ICUAS.2015.7152308

[18] Alessandro Giusti, Jerome Guzzi, Dan C. Ciresan, Fang-Lin He, Juan P. Rodriguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jurgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca M. Gambardella. 2016.

A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *IEEE Robotics and Automation Letters* 1, 2 (7 2016), 661–667. https://doi.org/10.1109/LRA.2015.2509024

[19] U. Hashmi, F. Afshan, and M. Rafiq. 2013. Performance Analysis of Different Optimal Path Planning Bug Algorithms on a Client Server Based Mobile Surveillance UGV. In *2013 4th International Conference on Intelligent Systems, Modelling and Simulation*. IEEE, 30–35. https://doi.org/10.1109/ISMS.2013.37

[20] Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. 2018. Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning. *IEEE Robotics and Automation Letters* 3, 3 (7 2018), 2539–2544. https://doi.org/10.1109/LRA.2018.2808368

[21] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. 2018. Deep Drone Racing: Learning Agile Flight in Dynamic Environments. (6 2018). http://arxiv.org/abs/1806.08548

[22] Tomas Krajník, Vojtech Vonásek, Daniel Fišer, and Jan Faigl. 2011. AR-Drone as a Platform for Robotic Research and Education. Springer, Berlin, Heidelberg, 172–186. https://doi.org/10.1007/978-3-642-21975-7{_}16

[23] Kedar Kulkarni and Aditya Bharadwaj. 2010. *Development of an autonomous aerial vehicle using Laser range finder.* Technical Report.

[24] Jeonghoon Kwak and Yunsick Sung. 2018. Autonomous UAV Flight Control for GPS-Based Navigation. *IEEE Access* 6 (2018), 37947–37955. https://doi.org/10.1109/ACCESS.2018.2854712

[25] Michael Leichtfried, Christoph Kaltenriner, Annette Mossel, and Hannes Kaufmann. 2013. Autonomous Flight using a Smartphone as On-Board Processing Unit in GPS-Denied Environments. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia - MoMM '13*. ACM Press, New York, New York, USA, 341–350. https://doi.org/10.1145/2536853.2536898

[26] Alexandros Lioulemes, Georgios Galatas, Vangelis Metsis, Gian Luca Mariottini, and Fillia Makedon. 2014. Safety challenges in using AR.Drone to collaborate with humans in indoor environments. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '14*. ACM Press, New York, New York, USA, 1–4. https://doi.org/10.1145/2674396.2674457

[27] Wenguang Mao, Zaiwei Zhang, Lili Qiu, Jian He, Yuchen Cui, and Sangki Yun. 2017. Indoor Follow Me Drone. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '17*. ACM Press, New York, New York, USA, 345–358. https://doi.org/10.1145/3081333.3081362

[28] Muhammad Atif Mehmood, Lars Kulik, and Egemen Tanin. 2008. Autonomous navigation of mobile agents using RFID-enabled space partitions. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems - GIS '08*. ACM Press, New York, New York, USA, 1. https://doi.org/10.1145/1463434.1463461

[29] Claudio E. Palazzi and Claudio E. 2015. Drone Indoor Self-Localization. In *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use - DroNet '15*. ACM Press, New York, New York, USA, 53–54. https://doi.org/10.1145/2750675.2750677

[30] Parrot. 2015. ARSDK Protocols Parrot SA. https://developer.parrot.com/docs/bebop/ARSDK_Protocols.pdf

[31] Roberto Sabatini, Alessandro Gardi, Subramanian Ramasamy, and Mark A. Richardson. 2014. A Laser Obstacle Warning and Avoidance system for Manned and Unmanned Aircraft. In *2014 IEEE Metrology for Aerospace (MetroAeroSpace)*. IEEE, 616–621. https://doi.org/10.1109/MetroAeroSpace.2014.6865998

[32] Fereshteh Sadeghi and Sergey Levine. 2016. CAD2RL: Real Single-Image Flight without a Single Real Image. (11 2016). http://arxiv.org/abs/1611.04201

[33] Mohammad Fattahi Sani and Ghader Karimian. 2017. Automatic navigation and landing of an indoor AR. drone quadrotor using ArUco marker and inertial sensors. In *2017 International Conference on Computer and Drone Applications (IConDA)*. IEEE, 102–107. https://doi.org/10.1109/ICONDA.2017.8270408

[34] Lucas Vago Santana, Alexandre Santos Brandao, and Mario Sarcinelli-Filho. 2015. An automatic flight control system for the AR.Drone quadrotor in outdoor environments. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE, 401–410. https://doi.org/10.1109/RED-UAS.2015.7441033

[35] Lucas Vago Santana, Alexandre Santos Brandao, and Mario Sarcinelli-Filho. 2015. Outdoor waypoint navigation with the AR.Drone quadrotor. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 303–311. https://doi.org/10.1109/ICUAS.2015.7152304

[36] Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Mike Elgersma. 2008. Flying Fast and Low Among Obstacles: Methodology and Experiments. *The International Journal of Robotics Research* 27, 5 (5 2008), 549–574. https://doi.org/10.1177/0278364908090949

An Exploratory Study into the Use of Autonomous Drones in Indoor Environments: Follow Me, Indoor Area Investigation and Indoor Navigation

Kasper Østergaard Helsted
Department of Computer Science
Aalborg University, Denmark
khelst13@student.aau.dk

Steffen Darby Carlsen
Department of Computer Science
Aalborg University, Denmark
sdca14@student.aau.dk

ABSTRACT

During the last 10-15 years, the world of robots has focused on the use of drones to explore areas inaccessible to humans, search/rescue and remote inspections. In this context, research has been made in regards to how autonomous drones should fly and how we interact with outdoor drones, but only limited work has focused on autonomous indoor drones and how we interact with these. In this paper, we present a 7 participant 3-parts exploratory study into autonomous indoor Human-Drone Interaction. Our tasks focused on primary interaction during navigation, voice interaction and secondary interaction. Our results show that participants expected the drone to behave under a level of *trust*. Trust was a metaphor expressed by some of the participants, they related this word to the behavior of the drone and how flawless they expected it to be. Our study outlines a series of design insights for future interaction development of indoor autonomous drones.

KEYWORDS

Drones, Human-Computer Interaction, Human-Drone Interaction, Indoor navigation, Indoor Interaction, Autonomous flight, Secondary interaction, Voice interaction, Bebop

1 INTRODUCTION

During the last 10-15 years, the world of robotics has largely focused on using teleoperated mobile robots equipped with cameras to get their eyes on something out of reach [25]. While in more recent times, drones has been used in regards to exploring areas completely inaccessible to humans[1], search/rescue [13] and remote inspections [22, 23]. In this context, researchers have proposed methods and approaches to autonomous outdoor drone navigation in regards to earthquakes[19], forest navigation[11] and tunnels [21, 22]. However, the area of indoor navigation and general use of autonomous indoor drones is rarely investigated. These fully autonomous drones interact differently with people and requires considerations and design decisions beyond the natural interaction techniques researched in regards to outdoor drones. The skill-set required for a human teleoperator of an indoor drone includes: Being able to control the drone in an indoor environment, avoiding collisions with surrounding obstacles and to interpret the intentions of other people in the indoor environment.

Given this, our work explores the use of drones in an indoor environment from a perspective of different types of interaction. Our results show that drones have a use in indoor environments, in particular in terms of indoor navigation. Furthermore, that interaction with drones requires *trust*, which can be achieved through the behaviour and flawlessness of the drone. Lastly, voice interaction with a drone performing micro movements requires the drone to understand advanced commands, while macro movements, such as moving between locations, were performed using similar commands to a GPS.

2 RELATED WORK

The interaction between humans and drones bring a new type interaction into the area of Human-Robot Interaction by adding an additional dimension. Being able to change the height of the robot interacted with, changes the interpretation of the interaction the user has. In the following, we will outline previous Human-Drone Interaction in regards to *appeal*, which Baytas et al. defined as: "Are people willing to accept, acquire, and/or use a drone or drones, as designed, for the purpose under investigation? Do people feel psychologically comfortable and safe in interacting with the drone(s) or simply cohabiting the same environment? Do they have confidence that the drone will not inflict damage or otherwise misbehave? [5]" Their definition of appeal and the context to drones is the primary factor investigated in this exploratory study. Thus we will, based on their definition, investigate the related work in this context. Lastly, interaction from the perspective of voice interaction is investigated in the related work.

2.1 Appeal

The differences between humans in regards to the level of comfort during interaction with drones are often associated with the impression of safety and stability, which are influenced by the design decisions of the drone. The understanding of what comforts people changes drastically depending on the background of the person [3, 12, 14]. Research has shown that drones meant for close range interaction should be small, as the noise and large amounts of airflow from the propellers can annoy the user causing discomfort [4, 6, 7].

The height of which a drone operates changes the comfort level of the users interacting with the drone. In a comparison between two heights, 120cm and 180cm, it was shown that participants preferred a height of 120cm [28]. Similar results was found by Han et al. who found that users allow a drone at eye to level closer than a drone above their heads [12].

Furthermore, the visual appearance of the drone is an important aspect of appeal. Different studies has investigated the impact of altering the visual appearance of the drone [16].

In contrast to the visual appearance of the drone, the movements of the drone has a big impact on the perception of comfort. Here,

studies have shown that fast mechanical and non-smoothed movements are repellent; while steady and smooth drone movements give a positive perception of the drone. Specifically, the undesired movements often includes lack of compensation of wind changes, causing the drone to perform unwanted and often drastic changes in short periods of time, while the positive perceptions come from smooth and stable movements, such as automatic movement pans [17, 24].

The last notion of *appeal* is privacy and social acceptance. This issue is highlighted by the Electronic Privacy Information Center, who highlights the issue of privacy and surveillance as a concern [2]. While many studies show that users are comfortable when interacting with drones, some users showed concern in how drones will integrate into public settings. Yao et al. interviewed drone pilots in the US, to understand their privacy perceptions and practices of drones [26]. They found that drone pilots consider privacy issues very important when they fly and consider the media to overstate drones as a privacy problem. This contradicts their own previous work, where they concluded that bystanders of drones were in search of different mechanisms to further privacy [27]. The results of bystanders showing a larger concern with regards to drone usage, is common within privacy research in regards to drones [7, 10, 15].

2.2 Voice interaction

Throughout the related work in regards to interaction, there is a fairly large body of work focusing on smartphone applications and joystick-based controllers. These are seen as the common interaction techniques with regard to drone piloting. In recent times, investigations and exploratory studies have focused on the use of gestures and voice control to communicate with drones [6, 8, 9, 20]. In the exploratory studies, it has been found that these two interaction techniques are the preferred techniques when interacting with social drones, with a relation to the interaction context and the personal familiarity of the participant with drones. Often, participants would display initial discomfort when interaction with drones using voice, but over time become comfortable with voice interactions and in some studies, where participants could choose between voice and gesture interactions, end up preferring to use voice, explaining that the interaction became similar to interacting with a pet [18].

3 EXPERIMENT

The contribution of this paper is a exploratory study into the world of autonomous drones using Wizard of Oz (WoZ). We performed three different parts with 7 participants, each providing some insight into design decisions for the future of HDI. Our aim was to emulate autonomous drone behavior in order to gain insight into how users choose to interact with a drone in indoor scenarios. In the following, we describe the three parts and and their results.

Indoor navigation
We simulated autonomous behaviour of a drone in an indoor environment. We chose an open lobby area as the initial start-off point, to perform a safe take-off and allow the user to better determine what they deemed a safe distance. The experimenter stayed behind the user, but could not be fully hidden, as they needed direct line of sight to the drone to properly control it. We found that with WoZ, users showed vastly different opinions in regards to safety and usefulness of an autonomous navigation drone. The scenario and task was explained verbally to user seconds before starting and the user was asked if they understood the task at hand.

Indoor area investigation
In order to test the interaction between a user and a voice controlled drone in an investigation scenario, two tasks were created. The user is told that they are sitting in their office, where they suddenly remember that they forgot their coffee machine and bag of coffee in another room. Instead of going to pick it up themselves, they send their drone to investigate and find them. Here they are to complete 2 tasks: Indoor navigation and Area investigation.

Secondary Interaction
In order to simulate a real-world scenario, the environment chosen mimics the worst case scenario. The environment chosen for the test, is a narrow hallway, where the user is asked to walk past the drone, while not doing anything in particular. The test was performed using WoZ. This was partly for safety reasons, since in case of a dangerous situation, the pilot would be able to abort the test. The pilot was not hidden, as they needed direct line of sight on the drone, in order to ensure safety of the participant. The scenario and task was explained verbally to the participant, to which the user was asked if they understood the task.

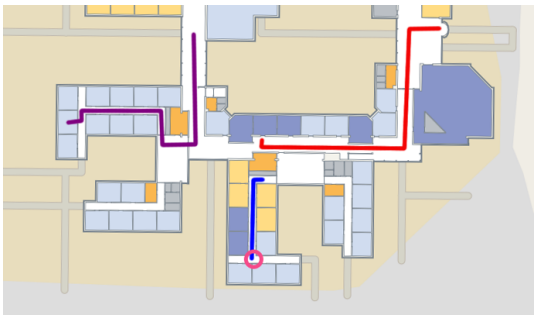

Figure 1: Indoor map displaying test areas, the red path is navigation test; the blue path is secondary interaction and the purple path is indoor area investigation

3.1 Participants

7 volunteers (6 male), 17 to 39 years old (μ = 27) were recruited from our institution and social networks. Their training was in Computer Science, Psychology, Social Sciences, shop trained and a High School student. All of the participants knew what drones were prior to the tests, three of the participants had previously flown a drone.

When testing the different parts of the experiment, we have chosen to use a limited amount of volunteers per part. For the first and third part, we are using 5 volunteers and for the second part we are using 2 volunteers.

3.2 Apparatus

In this section we describe the drone used for our experiment.

The drone used for our experiment is a Parrot Bebop 2 (42cm x 39cm x 9cm). In order to increase safety of our participants, we have modified the drone with propeller guards. The drone can be seen in Figure 2.

Figure 2: The Parrot Bebop 2 with mounted propeller guards

The propeller guards are mounted on the base of each propeller, and ensures that if the drone hits a wall or a person, the guard takes the blow, and not the propeller, thus minimizing damage to the drone and its surroundings. The drone has been limited in terms of speed and acceleration, these limitations are created for the drone to easier be controlled in limited space. The limitations set on the drone are the minimum in all directions, except yaw, where the recommended speed are used.

3.3 Procedure

The experiment lasted 30 minutes to 45 minutes. One of the experimenters performed all of the communication with the participant before the experiment and recorded during the experiment. The other experimenter controlled the drone and asked follow-up questions during the interview. The participant was told verbally about each of the tasks before the task started by reading the scenario aloud. They were asked if they understood the scenario and to ask the experimenters if they had any questions about the task. At this point the task would begin. After each task, the participant was interviewed about their interaction.

If the participant was partaking in the navigation and secondary interaction task, the order of which task they performed first was counterbalanced according to a Latin Square design.

3.4 Environment

Our experiment was carried out at the Computer Science building of Aalborg University, Cassiopeia. The experiment consisted of three different parts, indoor navigation, indoor area investigation a and secondary interaction.

The indoor navigation starts in the lobby of the building and moves towards room 0.1.32. The route is marked as the red path in Figure 1. This route was chosen since it starts from a location a user would be expected to start navigation from, and ends in an area where multiple turns has been taken. Lastly, the area flown in was also chosen due to being secluded from the group room clusters and having multiple doors at the ending location.

The indoor area investigation starts outside of a cluster and moves towards room 2.1.03, where the participant will perform the room investigation. The route is marked as the purple path, which can be seen in Figure 1. The route was chosen since it requires multiple voice commands in order to navigate and it allows participants to move the drone in all directions and through both wide and narrow areas.

Figure 3: The indoor area investigation and the object to be found

Inside of the room, the participant had to find two objects, a coffee machine and a coffee bag, which can be seen in Figure 3. We chose to hide the objects at different heights and to make them hard to spot in general. This was done to make the investigation of the area non-trivial and force the participant to control the drone in all dimensions. The areas where the two objects were hidden could not be found without changing the default height of the drone nor without actively navigating beyond the center of the room.

The last part, the secondary interaction was done in one of the hallways inside the Cassiopeia. The hallway measures 1.56m x 12m. The path is marked on Figure 1 with a blue path.

The participant starts at the end of the blue line, furthest away from the circle marked with red in Figure 1 and were asked to walk around the corner, assess the situation, and act as they would normally.

4 TASKS

In order to test the interaction with an autonomous drone in an indoor environment, a series of different tasks was created. Each task tests its own approach to indoor drone interaction with an autonomous drone.

Indoor navigation

In the indoor navigation task, the user has to be navigated from the entrance of a building to a target room. The user takes out their smartphone and starts the navigation of the drone. After the drone takes off, the user follows the drone in their preferred way.

Indoor area investigation

In the indoor area investigation task, the user has forgotten two objects at a previously visited location and wants to find these objects again. The user has connected to their remote voice controlled drone and wants to investigate the area.

First, the user has to tell the drone to perform a take-off, after which it will follow their commands, moving towards a group room. This part of the task requires the user to determine how they wish to control the drone when performing broad a navigational task, which can be seen as similar to a GPS giving you directions.

Second part of the task is after arriving at the target location, a group room. The user has to fly into the room and find two objects, a coffee bag and a coffee machine. These objects are hidden in such a manner that they can not be found without using all types of movements the drone can perform.

Secondary Interaction
In the secondary interaction task, the user has to interact with the drone in a narrow hallway under different scenarios. The user is asked to wait around a corner of the hallway and when their hear the drone taking off, count to 10 (allowing the experimenters to properly stabilize the drone and prepare it for the next scenario) before moving around the corner towards the end of the hallway. In the center of the hallway, the user will meet the drone in different scenarios, these being:

- On contact, land.
- Continue past the participant
- On contact, stop movement and await the participant to pass

These scenarios were performed at two different heights depending on the height of the participant, at their knees and at chest height. The starting height was counterbalanced according to a Latin Square design.

5 FINDINGS

In this section, we will first present the overall results of the experiment, to then go into specific aspects of the overall results of the exploratory study, by pointing out tendencies in the results and explaining these.

When following a drone, the participants expected the drone to know all possible scenarios and know how to properly react. This includes, always giving a 100% correct navigation, even in advanced buildings and to correctly handle larger crowds of humans in a navigational area. However, how the drone should handle larger crowds, none of the participants knew.

In correlation to being able to properly navigate crowded areas, the participants expressed the importance of the safety of any interactors with the drone. One of the repeated expectations of the drone was to detect and react *properly* to other humans during flight. In the navigation task, two participants said that trust is an important factor when using an autonomous system, like a drone. The participants expect the system to be flawless and always show correct directions and not misinterpret the environment it is flying in. This shows that using and interacting with a fully autonomous drone requires *trust*. This trust is based on the drone's ability to perform to its requirements and successfully interpret the intentions of the users. Whether this is for the drone to move faster, steer around on-going people or land. In regards to interpreting the intentions of people, the secondary interaction task showed that the participants preferred when the drone would move to the side and hold still, sharing the space similar to that of another person. In correlation to the general behaviour of the drone, the secondary interaction task showed that the participants preferred when the drone was at chest height. However, two of the participants disagreed with this, saying that the lower the drone would fly, the safer they felt. These two participants were also visible scared in the recordings of the task, showing that as users become more comfortable with the drone, the more they wish to have the drone in a similar height to how they usually interact with people.

When the participants where asked to interact with a drone using voice commands, two different types of commands were observed. The first type of interaction was a macro focused interaction. Here, the participants chose to use simple, general directional commands, such as; Move left, move right and move to the intersection. Considering that the participants were familiar with the environment and that there were only two intersections where they had to choose a direction, it makes sense that they choose to navigate using macro commands in this area. However, considering the wording using when explaining the tasks, explicitly telling them the room number they were flying to, it is surprising to us that none of them used the drone as a GPS, that they would tell where to move to, rather than telling it how to move.

After getting to the room, the participants had to find two objects. At this point, they changed their interaction to micro adjustments, telling the drone to *rotate slightly right*. Another noticeable difference was that they chose to, instead of telling the drone how to move, told it what to look at. An example of a command used by a participant was: "I want to take a closer look at the windowsill over there". This type of command show, that in order for a drone to properly cater to voice commands, it will need to have a certain level of understanding of natural language.

In general, the interactions with the follow-me navigation drone were simple and similar between the participants. They would choose their destination on the *application* and slowly follow behind the drone. During the straight paths of the drone's route, they would follow the drone relatively close, keeping up with the drone in terms of tempo. When at the intersection, they would prematurely slow down, keeping distance from the drone and waiting for the drone to manoeuvre the corner. It shows, that when the drone had to perform a different action than move straight, that users chose to keep their distance, waiting for the drone to finish its action.

The users expressed that a drone, depending on the environment it is used in, is a functional navigational tool. In particular, if the indoor environment is large enough and the infrastructure of the area needing navigation allows for it. However, there are certain expectations to the abilities and behaviour of the drone. The drone should follow the pace of the follower, while not exceeding a regular walking pace. Furthermore, it should be able to handle the environment and any changes that happen to it during the navigation. This includes meeting other people and moving objects.

6 DISCUSSION

We made an exploratory study into the world of autonomous drone flight by deploying a series of different WoZ tasks. The tasks focus primarily on the interaction between a user and a working autonomous drone in a navigation context. Throughout this study, we found trends in terms of the interaction techniques and expectations of drone behaviour from the feedback of the participants and

the analysis of video footage. This section presents and discusses design insights based on these trends.

6.1 Drone behaviour

Several different expectations in terms of drone behaviour was observed through the interviews and task recordings. We observed that users expected the drone to act under a level of trust and that the height of the drone and interaction action is important to consider.

Trust

One of the repeating metaphors, said explicitly by two participants, was that the interaction with an indoor drone requires a high level of trust. We observed this through the interviews of the participants, where they expected the drone to react properly in all situations, where *situations* range from meeting people during indoor navigation to always correctly navigating the user to their destination. The last point of trust mentioned by the participants is in regards to the speed of the drone. They wanted consistent and predictable movement, that they could follow while walking. The system the participants searched for was a flawless, consistent and predictable system, which shows that if drones were to be used for indoor tasks, that the development of specialised drones, that only complete a singular task, although is very good at this particular task, is a potential future area of research. Similar results have been found by other researchers, who found that the noise and air-flow from the propellers can cause discomfort for the users [4, 6, 7]. In regards to the consistency and predictability of drone movements, related work found that users expect smooth and stable movements by the drone [17, 24]. This shows that even in different scenarios, the fundamentals of interactions with drones remain the same and that *trusting* a drone is an important aspect of ensuring a user is comfortable.

Height and safety

A variable modified during the secondary interaction task was height. The participants met the drone in a narrow hallway, where it would perform one of three actions at two different heights. Here, the participants preferred the drone to be at chest height, saying that although the drone is able to do more damage when at chest height, that being able to keep direct line of sight with the drone is more important than having the drone in the proximity of the least damaging area of the body. Two of the participants fully disagreed with this testimony, saying that they preferred the drone as low it could get. This shows that although our task showed a slight favour in regards to the drone being at chest height, that future research can further task with how users prefer to interact with a fully autonomous drone in indoor environments. Related work, primarily focusing on the concept of a social drone, have found that users feel more comfortable around a drone between 1.2m and eye level of the participant [12**?**]. Although our work used two different heights, it still shows some connection between height and feel of safety.

Interaction

The last behavioural expectation observed during the experiment is in regards to the action performed by the drone when approached by a person in a hallway. Here, our results show that 3 out of 5 participants preferred the drone to pull over and wait for them to pass the drone, while the last 2 wanted the drone to land. Expanding upon this task, by using areas of different sizes and a drone developed for indoor usage might yield more consistent results. This is largely because of the participants preferring the drone to hold still in the air, mentioning that the sound of the drone changed drastically when performing a landing, making it seem more scary. This could be counteracted by using a smaller form factor of a drone, perhaps designed for indoor use. The related work has shown that users prefer a drone in an indoor environment to be small, as a small drone generates less noise and air [4, 6, 7].

6.2 Voice interaction

Two types of interaction methods were used by the participants during the voice interaction tasks. This was either macro or micro adjustments of the drone. The type of command changed how the user interacted with the drone, going from general and vague commands to specific, object targeted commands.

Natural language understanding

Macro commands were used by the participants when moving between locations, the commands consisted of simple directional commands, similar to how a GPS proposes directions. During this interaction, the drone was never told how to move, rather which direction it should move. This was changed after entering the room, where the commands changed into how the drone should move or commands specifying where the drone should look. This shows, that if the drone needs to perform micro adjustments, a greater level of natural language processing is required in order for a user to feel understood in their interaction with the drone. On the other hand, the simplicity of the directional commands makes it easier to have a drone perform navigational tasks using voice commands. One of the comparisons used in regards to drones, is that it is similar to that of a pet during interaction [18]. This shows that the level of understanding necessary to properly control a drone in an indoor environment, can be compared to the same communication level of interaction with a pet. This can, in future work, be used as a guideline of how much natural language understanding is necessary in the context of voice interaction with a drone.

7 CONCLUSION

Given the current increase in research in regards to fully autonomous indoor drones, we need to create and investigate natural interaction techniques to best support users. We executed one exploratory experiment with 7 users performing three different tasks and found design principles with high agreement among participants. We found several similar ties between participants in their expectations to how they interact with a drone, both when directly interacting and during secondary interaction. We contribute a set of design insights to develop the field of autonomous indoor Human-Drone Interaction.

In the future, there are primarily two points of interest based on our research. The first is to primarily focus on the areas where our exploratory study showed to be statistical insignificant. This is especially in regards to the preferred height of the drone, where our study showed mixed results between knee and chest height. The second point of interest is to use the proposed design insights and use those in the development of an autonomous indoor flight system for drones. The understanding of how a user expects a drone to behave and interact with an user, both in regards to direct interaction, during navigation, and secondary interaction, can be used to design a future system that is not only able to fulfil its goal, but also designed for human interaction.

Due to the complications of flying in confined spaces, some of the participants noticed that an experimenter flew the drone, rather than an automated system. We believe that this might cause some of the participants to feel safer than they really were. In order to further investigate and validate the results of our study, recreating our experiment with a fully autonomous drone might yield different results compared to ours.

REFERENCES

[1] [n. d.]. Drones help write new history of Caribbean – ScienceDaily. https://www.sciencedaily.com/releases/2016/10/161017123746.htm

[2] [n. d.]. EPIC - Domestic Unmanned Aerial Vehicles (UAVs) and Drones. https://epic.org/privacy/drones/

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

[4] Mauro Avila Soto, Markus Funk, Matthias Hoppe, Robin Boldt, Katrin Wolf, and Niels Henze. 2017. DroneNavigator. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '17*. ACM Press, New York, New York, USA, 300–304. https://doi.org/10.1145/3132525.3132556

[5] Mehmet Aydin Baytas, Damla Çay, Yuchong Zhang, Mohammad Obaid, Asim Evren Yantaç, and Morten Fjeld. 2019. The Design of Social Drones. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, New York, New York, USA, 1–13. https://doi.org/10.1145/3290605.3300480

[6] Jessica R. Cauchard, Jane L. E, Kevin Y. Zhai, and James A. Landay. 2015. Drone and me. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15*. ACM Press, New York, New York, USA, 361–365. https://doi.org/10.1145/2750858.2805823

[7] Victoria Chang, Pramod Chundury, and Marshini Chetty. 2017. Spiders in the Sky. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 6765–6776. https://doi.org/10.1145/3025453.3025632

[8] Chien-Fang Chen, Kang-Ping Liu, and Neng-Hao Yu. 2015. Exploring interaction modalities for a selfie drone. In *SIGGRAPH Asia 2015 Posters on - SA '15*. ACM Press, New York, New York, USA, 1–2. https://doi.org/10.1145/2820926.2820965

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

[10] Markus Funk and Markus. 2018. Human-drone interaction let's get ready for flying user interfaces! *Interactions* 25, 3 (4 2018), 78–81. https://doi.org/10.1145/3194317

[11] Alessandro Giusti, Jerome Guzzi, Dan C. Ciresan, Fang-Lin He, Juan P. Rodriguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jurgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca M. Gambardella. 2016. A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *IEEE Robotics and Automation Letters* 1, 2 (7 2016), 661–667. https://doi.org/10.1109/LRA.2015.2509024

[12] Jeonghye Han and Ilhan Bae. 2018. Social Proxemics of Human-Drone Interaction. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction - HRI '18*. ACM Press, New York, New York, USA, 376–376. https://doi.org/10.1145/3173386.3177527

[13] Yao-Hua Ho, Yu-Ren Chen, and Ling-Jyh Chen. 2015. Krypto. In *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use - DroNet '15*. ACM Press, New York, New York, USA, 3–8. https://doi.org/10.1145/2750675.2750684

[14] Walther Jensen, Simon Hansen, and Hendrik Knoche. 2018. Knowing You, Seeing Me. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1–12. https://doi.org/10.1145/3173574.3173939

[15] Brennan Jones, Kody Dillman, Richard Tang, Anthony Tang, Ehud Sharlin, Lora Oehlberg, Carman Neustaedter, and Scott Bateman. 2016. Elevating Communication, Collaboration, and Shared Experiences in Mobile Video through Drones. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems - DIS '16*. ACM Press, New York, New York, USA, 1123–1135. https://doi.org/10.1145/2901790.2901847

[16] 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. In *Proceedings of the 5th International Conference on Human Agent Interaction - HAI '17*. ACM Press, New York, New York, USA, 89–96. https://doi.org/10.1145/3125739.3125774

[17] Bomyeong Kim, Hyun Young Kim, and Jinwoo Kim. 2016. Getting home safely with drone. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct - UbiComp '16*. ACM Press, New York, New York, USA, 117–120. https://doi.org/10.1145/2968219.2971426

[18] Hyun Young Kim, Bomyeong Kim, and Jinwoo Kim. 2016. The Naughty Drone. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication - IMCOM '16*. ACM Press, New York, New York, USA, 1–6. https://doi.org/10.1145/2857546.2857639

[19] Tomas Krajník, Vojtech Vonásek, Daniel Fišer, and Jan Faigl. 2011. AR-Drone as a Platform for Robotic Research and Education. Springer, Berlin, Heidelberg, 172–186. https://doi.org/10.1007/978-3-642-21975-7{_}16

[20] Mohammad Obaid, Felix Kistler, Gabriele Kasparavičiūtė, Asim Evren Yantaç, and Morten Fjeld. 2016. How would you gesture navigate a drone?. In *Proceedings of the 20th International Academic Mindtrek Conference on - AcademicMindtrek '16*. ACM Press, New York, New York, USA, 113–121. https://doi.org/10.1145/2994310.2994348

[21] Tolga Ozaslan, Giuseppe Loianno, James Keller, Camillo J. Taylor, Vijay Kumar, Jennifer M. Wozencraft, and Thomas Hood. 2017. Autonomous Navigation and Mapping for Inspection of Penstocks and Tunnels With MAVs. *IEEE Robotics and Automation Letters* 2, 3 (7 2017), 1740–1747. https://doi.org/10.1109/LRA.2017.2699790

[22] Tolga Özaslan, Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. 2015. Inspection of Penstocks and Featureless Tunnel-like Environments Using Micro UAVs. Springer, Cham, 123–136. https://doi.org/10.1007/978-3-319-07488-7{_}9

[23] Chathura Suduwella, Akarshani Amarasinghe, Lasith Niroshan, Charith Elvitigala, Kasun De Zoysa, and Chamath Keppetiyagama. 2017. Identifying Mosquito Breeding Sites via Drone Images. In *Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications - DroNet '17*. ACM Press, New York, New York, USA, 27–30. https://doi.org/10.1145/3086439.3086442

[24] Daniel Szafir, Bilge Mutlu, and Terrence Fong. 2014. Communication of intent in assistive free flyers. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction - HRI '14*. ACM Press, New York, New York, USA, 358–365. https://doi.org/10.1145/2559636.2559672

[25] Waypoint Robotics. 2019. What Are Autonomous Robots? https://waypointrobotics.com/blog/what-autonomous-robots/

[26] Yaxing Yao, Huichuan Xia, Yun Huang, and Yang Wang. 2017. Free to Fly in Public Spaces. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 6789–6793. https://doi.org/10.1145/3025453.3026049

[27] Yaxing Yao, Huichuan Xia, Yun Huang, and Yang Wang. 2017. Privacy Mechanisms for Drones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 6777–6788. https://doi.org/10.1145/3025453.3025907

[28] Alexander Yeh, Photchara Ratsamee, Kiyoshi Kiyokawa, Yuki Uranishi, Tomohiro Mashita, Haruo Takemura, Morten Fjeld, and Mohammad Obaid. 2017. Exploring Proxemics for Human-Drone Interaction. In *Proceedings of the 5th International Conference on Human Agent Interaction - HAI '17*. ACM Press, New York, New York, USA, 81–88. https://doi.org/10.1145/3125739.3125773