# 2022 年臺灣國際科學展覽會
# 優勝作品專輯

|  |  |
|---|---|
| 作品編號 | 190043 |
| 參展科別 | 電腦科學與資訊工程 |
| 作品名稱 | Development of an autonomous Search and Rescue Drone |
| 得獎獎項 |  |

|  |  |
|---|---|
| 國　　家 | Switzerland |
| 就讀學校 | Berufs- und Weiterbildungszentrum Buchs |
| 指導教師 |  |
| 作者姓名 | Dominic Rinderer |

關鍵詞　Drone,　Search and Rescue

# 作者照片

# 1 Foreword

Nowadays, news services report more and more about environmental disasters. People must be rescued from forest fires or floods. Emergency forces can no longer keep up with evacuating and rescuing people. After the environmental disaster the search for survivors goes on for weeks. Helicopters fly non-stop, rescue workers fight their way through the rubble. And with the climate getting worse and worse, there's no end in sight. That's why, I thought about what I could do to make the rescuers' work easier.

From an early age I have been fascinated by flying objects of all kinds. At the age of eight I got my first RC helicopter and since then RC flying is one of my passions. At the beginning of my apprenticeship as a computer scientist I discovered programming. I started reading up on the field of Artificial Intelligence. Unfortunately, I never got to implement a larger project with an AI (artificial intelligence).

I will try to combine these two technologies - AI and drone. The goal of this project is to develop an autonomous Search and Rescue (SAR) drone. I will plan the software and implement it accordingly.

The Search and Rescue drone is designed to fly over a predefined area in the event of a natural disaster. In this area, it autonomously searches for survivors who need to be rescued. Compared to conventional search and rescue missions with helicopters, this has many advantages. The biggest of these is the difference in cost.

This paper will first cover the theory and then the implementation, functionality, and difficulties I had during the implementation.

## 2 Technology drone

On a beautiful, sunny day, you walk along a large lake. The birds are chirping, the cool wind is blowing through your hair, the children are playing on the shore. Suddenly, a loud whirring noise shatters the pleasant atmosphere. It's a drone! Twenty years ago, this scenario would have been unlikely. In recent years, there has been a huge hype towards drones. Nowadays you see them everywhere.

Drones are flight systems that can fly without a pilot. The pilot controls the drone from a distance. Originally, drones were developed for the military. They served mainly as target drones in anti-aircraft exercises. That way, no one came to harm. Today, things are different. Drones are used in a wide variety of applications. Among others, for intelligence, police, civil/ commercial or scientific purposes. (Unmanned Aerial Vehicle, 2019)

## 3 Search and Rescue drones

In recent years, drones have become integrated into the Search and Rescue sector. Drones can be equipped or expanded in many ways. For example, by adding cameras or sensors. These allow the drone to have an overview from above and collect data that can be analyzed to gain important information. Such systems are accessible to many user groups at low cost. For example, search missions using drones equipped with thermal imaging cameras are less costly than helicopter missions. In addition, helicopters are limited by their flight altitude. Drones can fly lower and accordingly reach places that are inaccessible to helicopters.

Drones help police investigate crime scenes or search for criminals. Firefighters use drones to analyze fires and to search for missing persons. They are often used in areas that are difficult for humans to reach. This is also the case, for example, at the North Sea and Baltic Sea. There, the German Lifesaving Society uses drones to save people from drowning. The drone can locate swimmers with a thermal imaging camera and drop rescue buoys. In the future, the drone of the startup "Bluebird Mountain" will speed up the search for buried avalanche victims and thus save lives. The drone is thrown into the air by the skier or snowboarder as soon as an avalanche starts behind him. The drone then follows the rider's avalanche transceiver and circles above the accident site as soon as he is buried. (These are the drones of the future, 2019)

### 3.1 Study on drones in rescue operations

In September 2018, a study was conducted to see if rescue workers could use drones to find victims faster. The study sent randomly selected teams of searchers to find simulated victims in the rocky fields and cliffs of Ireland and Wales. Thirty teams used commercially available drones with the standard built-in cameras, while the other 20 teams searched for victims on foot. Only 17 of the ground teams found their victims. In comparison, 23 drone teams found their victims. These numbers indicate that Search and Rescue protocols using drones are not advanced enough to take maximum advantage of drone technology. However, drone teams found their victims an average of 191 seconds faster. That's more than 3 minutes!

*Figure 1: Study on drones in rescue operations*

Searchers in the study said that finding victims with the drone was much more difficult than they would have expected. This shows that it is important to develop new procedures in this area: What flight patterns should be flown? What altitude provides the best coverage? What sensors are best suited to detect missing persons? Which areas are best searched by ground troops and which by drones? Answering these questions won't be easy, but it will have a powerful impact. (Drone Efficacy Study, 2018)

## 3.2 SAR drones on the market

Nowadays, there are many different drones designed for search and rescue missions. Some are even autonomous or have expensive infrared cameras. Below are some drones that are now available on the market.

### 3.2.1 Nokia Drone Networks

Nokia Drone Networks, based on Nokia Digital Automation Cloud, is a solution that includes Nokia drones, private and secure mobile broadband, cloud connectivity and a control center. With this solution, a fleet of drones can fly automated missions controlled from the command-and-control center. Data and information are collected in the process, for example, to meet business needs related to security and transportation, and to facilitate operations in mission-critical situations such as public safety. The drones are connected via a private mobile broadband network to ensure they are not affected by congestion on the public network, and manual operation is also possible if required.

*Figure 2: Nokia Drone Networks*

Nokia drones can be equipped or expanded in many ways, for example by adding cameras or sensors. The drone can also be equipped with a built-in camera as well as speakers, searchlights, customizable sensors for smoke, motion, radiation and much more.

This solution from Nokia is the most advanced on the market so far. (Nokia Drone Networks, no date)

### 3.2.2 DroneSAR

DroneSAR is not a drone, but a software. The app can be installed on any iPad. It connects to commercially available DJI drones, such as the DJI Mavic Pro. This drone is then controlled by the app during a Search and Rescue mission. With the app it is possible to define flight routes, mark and share important mission points, stream the position and live video of the drone over the internet. (Drone Software that saves lives, 2018)

This app is very similar to my project, with the difference that my navigation system is not as sophisticated, and my software can't share the video data over the Internet. But my system has built in image recognition.

### 3.2.3 DJI - M200 Series

This series of drones is the latest form DJI. They are specifically designed for industrial applications. The drone is resistant to rain, has stereo vision systems, runs on a two-battery system, and even has a modular expansion port to connect your own hardware. The drone is foldable like other models from DJI. The flight time is 35 minutes, and the maximum payload is 2 kilograms. The special thing about the drone is that it has a dual-gimbal system. This means that up to two

*Image 3: DJI Matrice 200 Series V2*

cameras can be used at the same time. This allows you to use a special camera with optical zoom and an infrared camera at the same time. This is ideal for search and rescue missions. Like all DJI drones, the drone has a collision avoidance system. This allows the pilot to concentrate on the mission. (Matrice 200 Series, 2019)

This drone is better than the drone I have. However, it needs to be flown by a pilot. It also does not have built-in image recognition software.

## 4 Interviews

I have conducted interviews in the USA and Switzerland about search and rescue drones.

### 4.1 Interview with Buchs Fire Department

In Buchs, I conducted an interview with Marcel Senn, the commander of the Buchs Fire Department. I asked him questions regarding drone operations at the Buchs Fire Department:

**How long has the Buchs Fire Department had the drone?**

We have been using a drone since 2016.

**What kind of drone are we talking about?**

The Buchs Fire Department has a DJI Phantom 3 Pro.

**Under what circumstances is the drone used?**

The drone is used in chemical spills and fire incidents.

**Is the drone helpful?**

Yes, you can get a quick and safe overview during chemical spills and fire operations.

**How often is the drone used?**

Rather rare, as chemical spills don't happen very often.

**How does a drone mission work?**

A dedicated drone pilot flies the drone mission according to the wishes of the incident commander. The pilot stands next to the incident commander and thus has an overview of the situation.

**Does the drone need special maintenance?**

No, it does not need any special maintenance. Only the batteries need to be charged after each use.

### 4.2 Interview with Woodville Fire Department

I was in the US for the fall holidays visiting family. While there, I had the opportunity to ask the commander of the fire department for an interview at a charity event. Delighted, he accepted. Woodville is a small village in the north-east of the USA. It is in the state of Ohio.

*Image 4: Woodville Fire Department*

The Woodville Fire Department does not have its own drone. It regularly borrows the drone from the Woodville Police Department. However, the surrounding fire stations have their own drones. The Woodville Fire Department is not very large. The drone is operated by a specially trained firefighter. It is mainly used to see the extent of a fire or to survey the surrounding area to plan the best strategy to fight the fire. The drone is mainly used to gather information. It is also used in the search for missing persons.

While I was in the US, an elderly woman suffering from dementia went missing. The Woodville Fire Department had three drones out searching for the woman. One of the three drones was a very expensive one, specially designed for Search and Rescue missions. It even had infrared and night vision. Thanks to the drones, the woman was found in a remote forest. Without this technology, the woman might have remained missing forever. After this incident, the Woodville Fire Department decided to get a drone of their own.

# 5 Artificial intelligence

Artificial intelligence is a branch of computer science that deals with the automation of intelligent behavior and machine learning. Artificial intelligence is not about real intelligence. Rather, it attempts to simulate "intelligent behavior" through simple algorithms. In this process, a computer is built and programmed in such a way that it can process problems relatively independently. (Coastal Intelligence, 2019)

## 5.1 Haar-cascade detection

Object recognition using Haar feature-based cascade classifiers is an effective object recognition method invented by Paul Viola and Michael Jones. It is a machine learning based approach where a cascade feature is trained from many positive and negative images. It is then used to recognize the same objects in other images.

Paul Viola is a computer vision researcher, former MIT professor and vice president of science for Amazon Air. For his work on the Haar-cascade object recognition method, he received the Marr Prize in 2003 and the Helmholtz Prize from the International Conference on Computer Vision in 2013. Michael Jones is also a computer vision researcher. He won the Marr Prize with Paul Viola and later the CVPR Longuet-Higgins Prize.

To train the classifier, many positive images are needed. These contain the object that should be recognized. Many negative images are also needed. These do not contain the object. Then, using the Haar features shown below, different features are detected. Each detected feature is a single value, calculated by subtracting the sum of the pixels under the white rectangles from the sum of the pixels under the black rectangles:

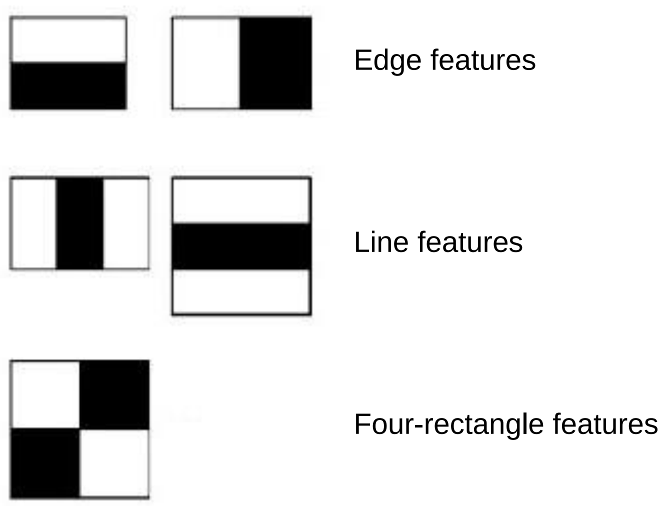


*Image 5: Hair features*

These features are then used to categorize smaller sections of the image. The human face serves as an example. In all human faces, it is common for the region of the eyes to be slightly darker than the cheeks. Therefore, a common Haar feature for face recognition is a set of two adjacent rectangles that lie over the eye and cheek region. The position of these rectangles is defined relative to a recognition window, which acts like a bounding box for the target. Here is an example of a human face:

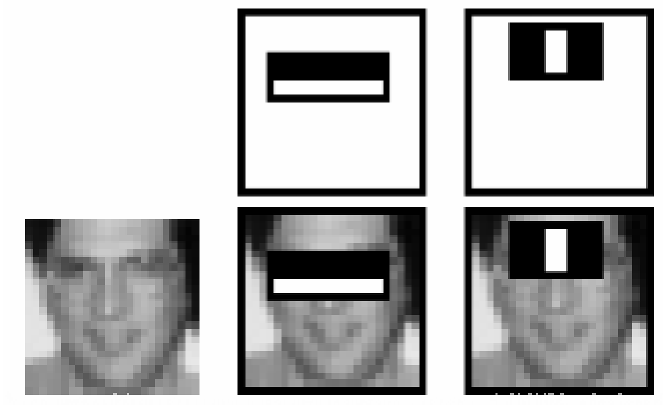


*Figure 6: Hair*

The great advantage of Haar-like features is the computational speed. Due to the use of integral images, a Haar-like feature of any size can be computed in constant time. Therefore, I used it for person detection as well. When flying by, the persons must be detected as fast as possible i.e., before the drone has flown past the person. (Cascade Classifier, 2019)

## 5.2 Deep learning based estimation of human pose

Pose estimation is a general problem in image recognition. It involves an attempt to estimate the position and orientation of an object. This is usually done by trying to detect and describe key points of the object. Until recently, there has been almost no progress in pose estimation due to the lack of good datasets. However, in recent years some good datasets have been published:

- COCO Keypoints challenge (COCO 2018 Keypoint Detection Task, 2018)
- MPII Human Pose Dataset (MPII Human Pose Dataset, 2018)
- VGG Pose Dataset (Human Psoe Evaluator Dataset, 2018)

The pose estimation model I used to distinguish help-seeking individuals from ordinary individuals is based on a paper called "Multi-Person Pose Estimation" (Pose Estimation, 2017) , written by the Perceptual Computing Lab at Carnegie Mellon University.

### 5.2.1 Architecture

The model takes as input an arbitrary image. This is processed, and as output you get the 2D positions of key points of each person in the image. This works in three stages:

Stage 0: The first ten layers of VGGNet are used to create feature maps for the input image.

Stage 1: This stage consists of two parts. In the first part, a multi-stage CNN is used to predict a set of 2D confidence maps of body part positions. For example, these body parts can be the elbow or the knee. The following image shows the confidence map for the left shoulder:

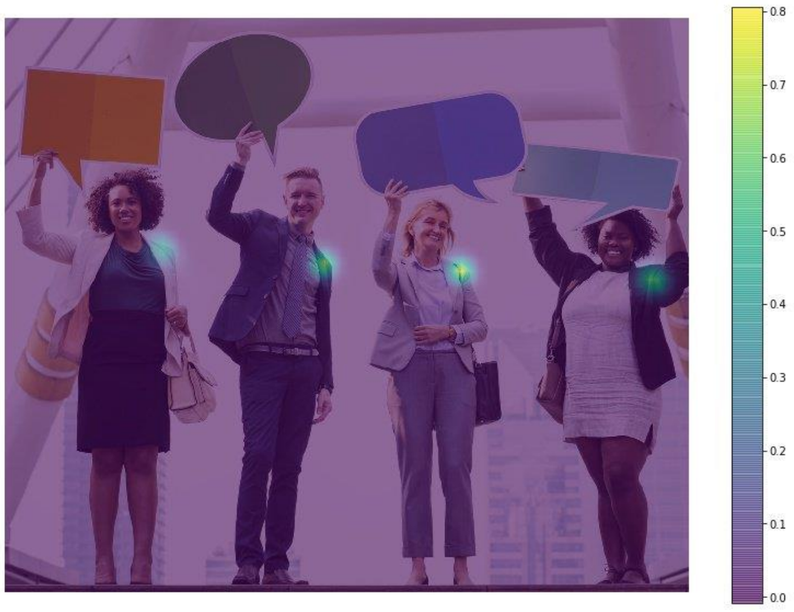


*Image 7: confidence-left-shoulder*

The second part of this stage attempts to predict 2D vector fields of partial relationships between key points. The figure below shows a partial affinity between neck and left shoulder:

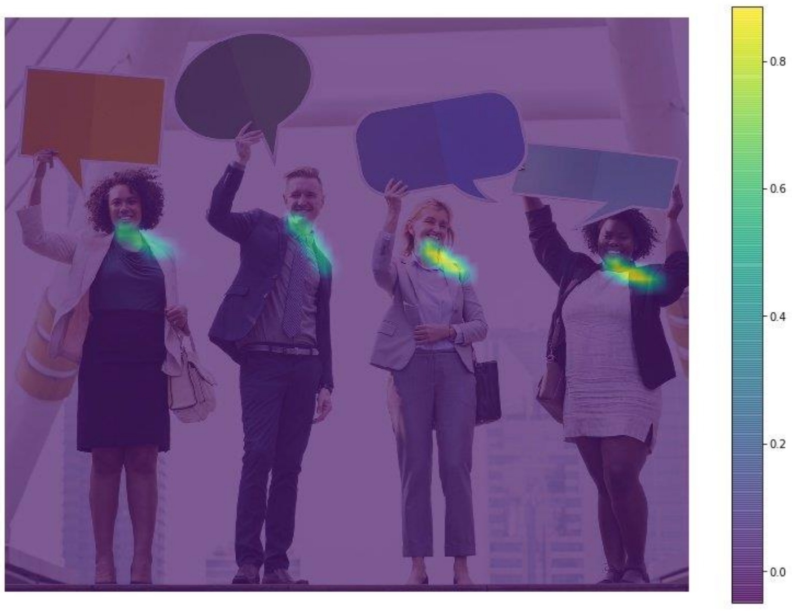


*Figure 8:heatmap-left-shoulder*

Stage 2: The trust and relationship maps are analyzed to create the 2D key points for all the people in the image. (Deep Learning based Human Pose Estimation, 2018)

# 6 Concept

The plan for this Project is to develop an autonomous Search and Rescue drone. Since my financial resources and time are limited, I am only writing the software. I am not building my own drone; therefore, I need to buy a drone as a base for this project.

My Search and Rescue drone is a prototype. I didn't have much time, so the solution is not the best. It is meant to show that the system I developed can work in principle. The idea is that it can fly over a predefined area in the event of a natural disaster. In this area it autonomously searches for survivors who need to be rescued. Compared to conventional search and rescue missions with helicopters, this has many advantages. Missions with a helicopter are very expensive. You buy a drone once and then have almost no further follow-up costs. A helicopter needs constant maintenance. It also needs specially trained pilots, which cost a lot.

# 7 Planning

The first thing I had to think about was what the autonomous Search and Rescue drone should be able to do. Accordingly, I defined goals:

## 7.1 Basic objectives

The aim is for the Search and Rescue drone to be autonomous, i.e. to be able to carry out a mission without human interaction. A mission consists of flying from a starting point to a predefined area. This area is then flown over by the drone. While flying over the area, the drone uses image recognition to search for survivors or those in need of rescue. It should be able to distinguish whether a person needs to be rescued or not. Rescuers do not need to be rescued. If it finds a person who needs to be rescued, this is reported to the operations center. That dispatch center can then send a helicopter to recover the person. Here is a summary of the basic objectives for the Search and Rescue drone:

- The drone can autonomously fly over an area

- The drone can detect humans
- The drone can distinguish between people seeking help and those not seeking help
- The drone can notify an operations center

## 7.2 Technical goals

Above I have defined the basic goals. Since this system is technically more complex and there are many intermediate steps, I distinguished between technical and basic goals. The technical goals are the following:

- Communication between drone and laptop works
- The navigation system works
    - Mission area can be defined
    - Validation of the mission area (In range, not too big?)
    - Drone can fly to mission area
    - Drone can efficiently fly over mission area
    - Drone can fly back to the starting point after the mission
- Autonomous flying works
- Mission can be cancelled manually
- Streaming the video data to the laptop works
- Person recognition works
- Distinguishing between those seeking help and those not seeking help works
    - Attitude of the person can be recognized
    - Recognize if arms are pointing upwards
- Message to operations center works
    - GPS position of the person can be transmitted

## 7.3 Programming language

The programming language I have been using is Python. I haven't implemented a real project with Python until now. That's why I see the autonomous Search and Rescue drone as a perfect project to learn Python. Python is a programming language available for free. It is considered to be particularly easy to read, which is related to the given structured programming style. It is very easy to download and use additional libraries. This makes Python a very powerful programming language. Also, I would like to program the whole software object based. I have never done this before. The big advantage of object-oriented programming is the reusability of the software components. This makes the code easier to read and more nicely structured, which increases the quality of the software in general. Furthermore, the software can be visualized using the UML notation. (Python, 2019)

*Image 9: Python Logo*

## 7.4 Drone

Next, I had to decide which drone to buy as a base for the Search and Rescue drone. I had the following technical requirements for the drone:

- Has a camera
- Interface for laptop
- Has GPS
- Can be used outdoors
- Minimum distance of one kilometer
- Not too expensive

I have decided to use the Parrot ANAFI. The following matrix justifies my decision:

| Drone model | Camera | Interface for laptop | GPS | Out-door | Dis-tance | Price |
|---|---|---|---|---|---|---|
| DJI Mavic Pro 2 (Mavic Pro, 2019) | Yes | Yes | Yes | Yes | 5'000 m | 1499.- |
| Ryze Tech Tello (Tello, 2019) | Yes | Yes | No | No | 100 m | 80.- |
| AML Pixhawk (Pixhawk, 2019) | No | Yes | No | No | 100 m | 195.- |
| Parrot ANAFI (ANAFI, 2019) | Yes | Yes | Yes | Yes | 4'000 m | 644.- |

Table 1: Evaluation drone

The DJI Mavic Pro 2 has no Python interface and is way too expensive. The Ryze Tech Tello has no GPS and is also not suitable for outdoor use. The Parrot ANAFI drone is just perfect. It is a bit expensive pricewise, but it is still within my budget. Moreover, it has a Python interface. This is the programming language I wanted to use for this project. Plus, it has a range of four kilometers. And the best is yet to come. There is a virtual environment for the Parrot ANAFI drone. This means I can immediately test the code I write on a virtual drone. This way, if I make a mistake, the real Parrot ANAFI drone won't get hurt. The virtual environment is called Sphinx. It simulates the real firmware of the drone, so it is identical to the real drone.

Image 10: Parrot ANAFI

## 7.5 Libraries

Libraries are collections of functions and methods. They provide additional functions, so that you don't have to write them yourself. There is no point in reinventing the wheel! I needed the following libraries for the autonomous Search and Rescue drone:

### 7.5.1 Olympe

The Olympe library was developed by Parrot. Olympe is a programming interface between Python and Parrot drones. It is required to connect to and control the drone via a remote Python program. Olympe is used to send commands from the Search and Rescue system to the drone.

However, Olympe is only the interface between the drone and Python, not between my Search and Rescue system and the drone. I had to program this myself. (Olypme Documentation, 2019)

### 7.5.2 OpenCV

OpenCV is the abbreviation for Open Source Computer Vision Library. OpenCV is an open-source computer vision and machine learning software library. I used OpenCV to implement person recognition and distinguishing between a person seeking help and a person not seeking help. (OpenCV, 2019)

### 7.5.3 NumPy

NumPy is the abbreviation for Numeric Python. NumPy is used to provide powerful data structures for efficient computation with large arrays and matrices. NumPy offers a huge number of high-quality mathematical functions, such as minimization, regression, fourier transformations, and more. (NumPy, 2019)

### 7.5.4 Pynput

This library allows to control and monitor various input devices. I use this library to read keyboard inputs. (pynput 1.4.5, 2019)

### 7.5.5 Geopy

Geopy is a library to make various calculations with GPS. I use it to calculate distances between two GPS coordinates. (GeoPy documentation, 2018)

### 7.5.6 Tkinter

Tkinter is a library to program a user interface with Python. I use this library for the notification to the rescue center. { "tkinter", 2019}

## 7.6 Structure

I divided the Search and Rescue System into different components and files. This way I could develop the components individually and finally merge them. In the chapter "Implementation" I will go into the different components and explain how they work. The Search and Rescue System is divided into the following components:

- Interface
- Navigation system
- Searchsystem
    - Person recognition
    - Distinction between persons seeking help and persons not seeking help
- Streaming
- Mission Abort System

These components were implemented in the following files:

| | |
|---|---|
| • Adrone.py: | Interface. |
| • Ai.py: | Person recognition and discrimination between people seeking help and people not seeking help. |
| • Mission.py: | This is the main file from which the whole system is started. |
| • Navigation.py: | Navigation system. |
| • Search.py: | Search system. |
| • Streaming.py: | Streaming the video of the drone. |
| • Settings.json: | Miscellaneous settings. |
| • Mission.json: | Definition of the mission area. |
| • Other files | |

## 7.7 GitLab

Anton Kiekels let me use his GitLab server for this project. Gitlab is a complete DevOps platform. I upload my newly written code to GitLab. This way I always have an extra backup. GitLab also automatically creates a versioning of the entire project. If I make a change that doesn't work, I can easily revert to an earlier, working version. In addition, versioning also allows me to track what I did and when. I can also work on the same project from different computers. The latest version of the project is always on the GitLab server.

# 8 Implementation

In this chapter I explain how the system works, how I implemented it and what problems I encountered. I won't explain every little detail, as I'm afraid it might get too technical and thus too complicated.

## 8.1 Olympe, Sphinx, Linux

Parrot-Sphinx is a simulation software to test the written code for Parrot drones. Sphinx simulates the firmware of a Parrot drone on the computer in an isolated environment. Gazebo is used as a base to simulate the physical and visual environment of the drone. I wrote the whole Search and Rescue system using Sphinx. Every little change I made I tested with Sphinx. This had the advantage that I didn't have to test the code on the real drone. Otherwise, it would have crashed into the ground quite often. This also saved me a lot of time, as I didn't always have to go outside to test the code I had written. (Sphinx, 2019)

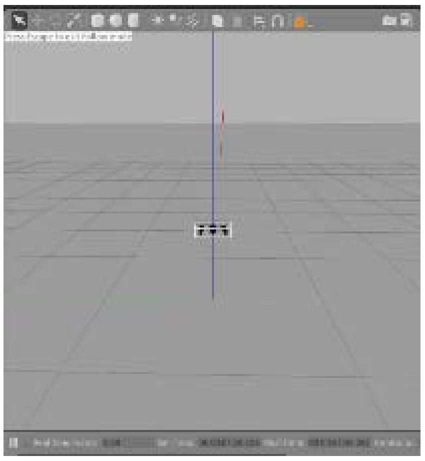


Image 11:Sphinx

Sphinx and Olympe unfortunately only run on the Linux operating system. This means I had to dig out my old laptop and reinstall it with Linux. Installing Olympe and Sphinx was easy once I figured out how to do it. You see, at first, I wanted to run the firmware of the ANAFI drone on my server. However, this was a bad idea as it would have made the whole thing way too complicated. Also, there is no advantage of running the firmware on the server. That's why I finally decided to install everything on the laptop.

## 8.2 Interface

All the code for the interface is in the file called adrone.py. All commands that go to the drone are sent through the interface. The interface is the only part of the system that can communicate with the drone. An example of this: The navigation system calculates what direction is the most efficient direction to fly. The drone must then align itself accordingly and fly an offset. The navigation system passes this information to the interface, which then translates the information into commands that can be executed by the drone.

## 8.3 Navigation system

The navigation system has the following tasks:

- A mission area can be defined via the navigation system.
- It calculates if the mission area is within a predefined distance from the start position.
- It calculates how many times the drone must fly back and forth until the entire mission area has been flown over.

### 8.3.1 Define mission area

The mission area is defined in a json file. (JavaScript Object Notation, 2019) It works by specifying a GPS coordinate. This GPS coordinate is one of the four corner points of a field. Only rectangles can be flown over. If I would have solved this differently, I would have had to invest more time, which I simply did not have. Once the GPS coordinate is defined, the whole thing works like a coordinate system. The GPS coordinate is the center of the coordinate system. So, you specify the distance on the X-axis and the Y-axis in meters. This can be positive or negative. If an axis is negative, then the GPS coordinate is simply another corner. The X-axis always points to the north. However, an angle can be specified. This angle is the deviation of the X-axis to north. The angle can also be negative. Using this system, it's possible to define a rectangle in every possible shape and direction. The content of the json file looks like this:

```
{
    { "area".
        }, "coordinate": {
            "latitude": 48.879190525815225,
            "longitude": 2.368439865681848
        },
        "x": 100,
        "y": 120,
        "degrees": 0
    }
}
```

"Area" is the object that contains all the information. "Coordinate" is the coordinate with the latitude and longitude. X and Y are the axes and "degrees" is the deviation from north in degrees. The settings described above look graphically as follows:

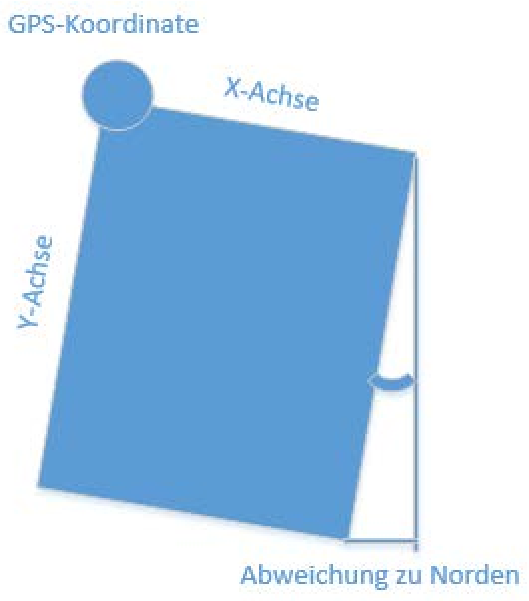


*Figure 12: Mission area*

## 8.3.2 Distance calculation of the mission area

Another part of the navigation system is to check if the defined mission area is within the range of the drone. The maximum range of the drone can be defined in the settings (settings.json). First, the current position of the drone is determined. Then with Geopy (GeoPy documentation, 2018) the distance of the two GPS coordinates is calculated. The calculations are different depending on which corner point the specified GPS coordinate is. This is checked in the navigation system. The corners of the mission area are named. Dc1 means "distance to corner 1", which is the distance between the start position of the drone and the lower, left corner. The corners are numbered in a clockwise direction. In the above example, the calculation looks like this, where in this example dc2 is the distance between the corner point and the position of the drone:

$$dc3 = \frac{x}{\sin\left(\tan^{-1}\frac{x}{dc2}\right)}$$

$$dc4 = \frac{-1 \times y}{\sin\left(\tan^{-1}\left(\frac{-1 \times y}{dc3}\right)\right)}$$

$$dc1 = \frac{x}{\sin\left(\tan^{-1}\left(\frac{x}{dc2}\right)\right)}$$

The greatest distance of the calculated distances must not exceed the maximum range of the drone.

### 8.3.3 Overfly mission area

This part of the navigation system calculates which is the most efficient flight direction and at what angle to north the drone must fly accordingly. The most efficient flight direction is the one in the direction of the longer axis. So, in the above example, it is more efficient to fly along the Y axis. Otherwise, the drone would have to stop more often to turn around. The drone is flying 15 meters above the ground. The camera is tilted down 45 degrees. Then the drone sees about 50 meters of the ground. The whole thing can be represented graphically:

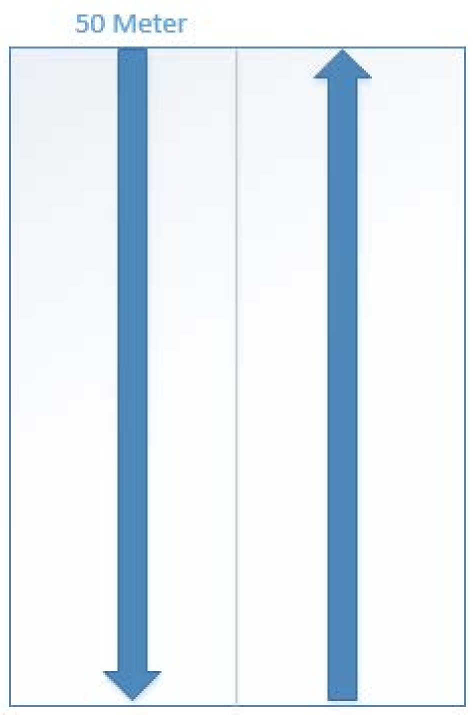


*Figure 13: Fly over mission area*

## 8.4 Searchsystem

The Search system is turned on once the drone reaches the search area and has aligned itself. It is the part of the Search and Rescue System that searches for the actual people that need to be rescued. The Search System has the following tasks:

- Switch on streaming and align camera
- Control airspeed
- Person recognition
- Distinguish between people seeking help and people not seeking help
- Flight maneuvers for found persons
- Message to operations center

### 8.4.1 Switch on streaming and align camera

Once the search area is reached, a command is sent to the drone to turn on the camera. The image is then transmitted to the laptop. There, the image can be processed further. The drone's camera is tilted downwards by 45 degrees. This is the best way to search for people. I have tested this extensively. More about this is covered in the chapter Streaming.

## 8.4.2 Control airspeed

There is a command "moveBy" for the drone. The drone flies a given distance in one direction. I wanted to use this command for flying over the search area. However, the speed can't be controlled so easily. There is a command called "setAutonomous-FlightMaxHorizontalSpeed". However, this was always ignored by the drone. I just couldn't figure out why. One forum talked about a bug in the drone's firmware. If the speed is not specified, the drone just flies as fast as possible. This way people can't be detected. The solution to this problem was to use the command "PCMD" to fly forward. The command is used to simulate the joystick of a controller. This can regulate the speed by not pushing the joystick all the way forward. Now regulating the speed was possible. But I could not control the distance flown by the drone, so I had to think of a solution again.

It is possible to retrieve the speed of the drone. The speed in the north direction and south direction is returned by the drone. With these values I can calculate the current speed and direction of flight. Then I measure the time that passes while the drone is flying forward. This allows me to calculate the distance traveled. This is done about a hundred times a second. Then I just push the joystick forward until the desired distance is covered. The calculation looks like this, where Vn is the speed towards north, Ve is the speed towards east and $\Delta t$ is the elapsed time:

$$S = \Delta t \times \sqrt{Vn^2 + Ve^2}$$

## 8.4.3 Person recognition

I implemented the person detection using OpenCV. I used the Haar feature-based cascade object detection method for the person detection (Cascade Classifier, 2019) . In the theory section, I have explained how this method works in more detail. I used this method because it requires less power from the computer compared to other methods. Unfortunately, my laptop doesn't have enough power to use a better and more accurate method.

In order to train person recognition, I had to obtain positive and negative images. Here, the positive images contain the object I am trying to recognize. The negative images do not include the object. To procure the images I wrote a small program that would fly the drone at 15 meters and point the camera down at 45 degrees, stream the video from the drone to my laptop, and finally save it. After that, my family and I took some videos of us walking back and forth on different surfaces and waving our arms. From these videos, I was then able to export the individual frames using another program. Using OpenCV's annotation tool, I was able to mark each person on the frames. This process took many hours. When I was done, I had the positive images. After that, I had to take care of the negative images. I repeated the whole procedure with the difference that there were no people on the videos. Now, I also had the negative pictures. (Cascade Classifier Training, 2019)

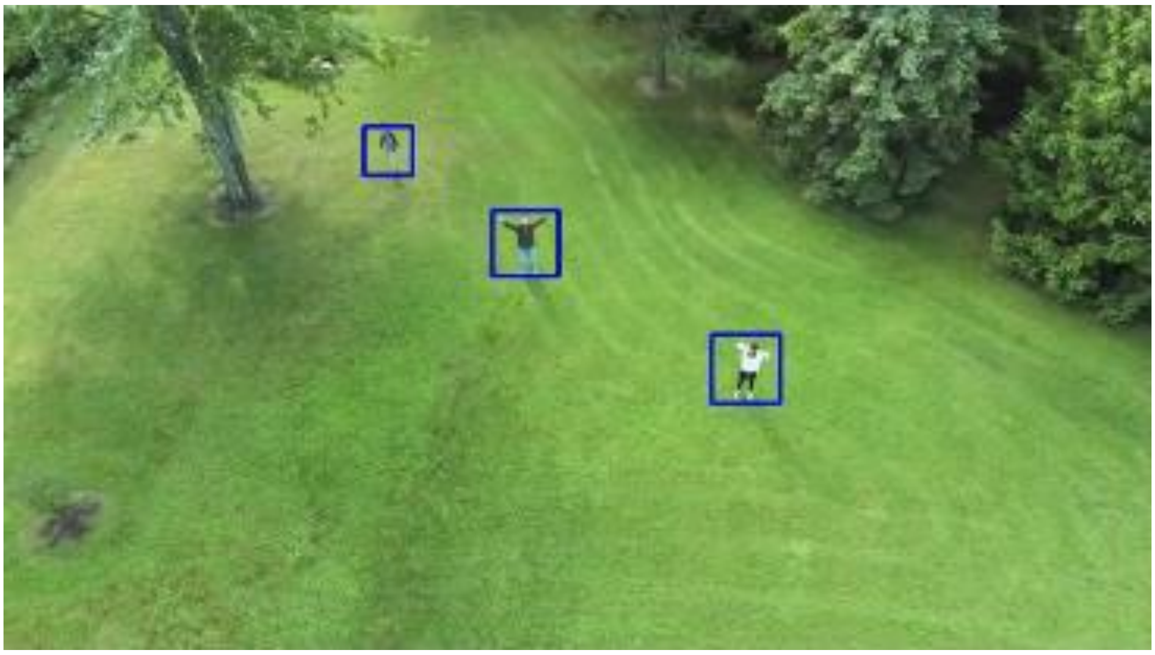


*Figure 14: Person recognition*

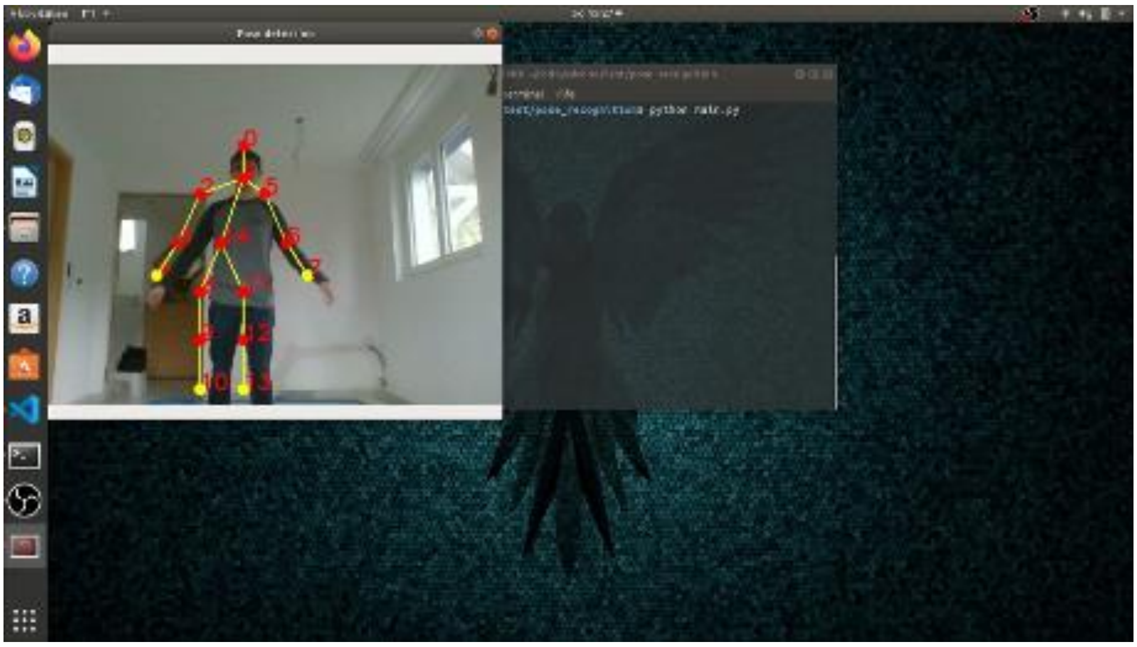
The actual training of the person recognition was not very difficult, I just had to generate some files and make configurations. I tried to achieve an accuracy of about 98%, this goal was way too ambitious. The server on which the AI was trained was calculating for over two weeks! While the accuracy improved over time, each level took double the length of time to reach the next level.

Finally, I gave up because I would not achieve the desired accuracy before the end of the project. The final accuracy was about 50%, in every second image where there is a person, they were detected. This is not very accurate, but it can be improved with time.

### 8.4.4 Distinguish between people seeking help

In order to distinguish a person to be rescued from a "normal" person, there must be a sign or characteristic that can be recognized. Therefore, I assumed that a person to be rescued looks up at the drone and waves his arms.

For this, the pose of the person needs to be recognized. After that, the arms can be checked whether they are pointing upwards. For this, I used a pre-trained Deep Neural Network that estimates the pose of a person. Training such a neural network myself went beyond the scope of this project, the rest was straight forward. The neural network is based on OpenCV, so I only had to download it and import it. After that, the picture of the person is passed to the neural network and the position of important points of the human body is returned. A function then checks the angle between the shoulders and arms. If this angle is greater than 140 degrees, the arms are above the head and the person must be rescued (Tello Python, 2019) . It looks like this:



```
[illegible]/pose_recognition$ python main.py
```

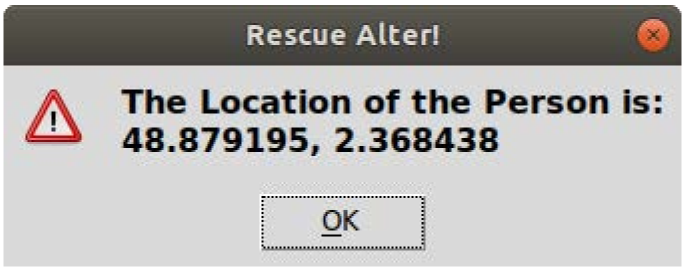
Figure 15: Pose detection

### 8.4.5 Flight maneuver

If a person is detected, the drone must stop and check whether or not the person needs help. This flight maneuver is performed by the search system. First, it verifies whether a person was really detected. Since the accuracy of the person detection is not very accurate, a person may be detected even though there isn't a person. Once the existence of a person is verified, the pose of the person is checked. Then, depending on the pose, a message is sent to the control center. Subsequently, the drone flies on and waits 10 seconds until it switches on the person detection again, otherwise the same person will be detected again and again.

### 8.4.6 Message to operations center

The problem of messaging the control center was solved using a simple popup message. To create the popup message, I used a library called Tkinter. The message contains the coordinates of the person to be rescued. (Tkinter, 2019) . The message looks like this:



Rescue Alter!

**The Location of the Person is:**
**48.879195, 2.368438**

Figure 16: Message to control center

## 8.5 Streaming

The streaming part of the Search and Rescue system transmits the live video from the drone to the laptop, it's then processed on the laptop. The problem is that the video must be received from the drone while the drone is also performing flight maneuvers. Therefore, these two activities have to be done simultaneously. In computer science the solution for this problem is called "multithreading". The different activities are executed on different cores of the processor. Thus, it is possible to receive and process the live video of the drone while the drone is executing flight maneuvers.

Problems also arose because of the multithreading. The activities run parallel, but somehow, they must be able to communicate with each other. The main system must be able to give the streaming system the command to start detecting people. And the main system in turn has to send a message to an operations center if a person is found. I solve this by setting so called flags. One of these flags is called "rescue_person". This is set to "True" as soon as the streaming system detects that a person needs to be rescued. The main system checks the value of this flag all the time. As soon as the main system recognizes that the flag has been set, it can initiate further actions. This allows the different systems to communicate with each other, despite multithreading. (An Intro to Threading in Python, 2019)

## 8.6 Mission Abort System

As the name of the mission abort system implies, it is used to abort a mission. The abort of the mission is initiated manually by pressing the A key on the keyboard, which stands for abort. Once the mission abort is initiated, the current flight maneuver of the drone is immediately interrupted. After that it returns to cruising altitude which is defined in the settings (settings.json). As soon as it has reached this altitude, it flies independently to the starting point and lands.

The mission abort system consists of two parts:

- Monitoring the keyboard inputs
- Flying the drone

### 8.6.1 Monitor keyboard inputs

Monitoring the keyboard inputs was very easy to implement. I use a library called "Pynput", it does all the work for me:

```
#Starting Keyboard listener
self. listener = keyboard. Listener(
    on_press=self. abort_mission)
self. listener. start()
```

Self.abort_mission is the function that is executed when a key is pressed. This function checks if the key that is pressed is the A key. If it is, the mission abort is triggered.

### 8.6.2 Threading problem

The mission abort system took over a week to implement. I was having problems and it took me a lot of time to find the bug. I programmed the mission abort system before the streaming system. Therefore, I had no experience with threading in Python at all. (An Intro to Threading in Python, 2019) . Honestly, I didn't even know what it was. The symptoms of the problem were that the drone seemed to fly around randomly when the mission abort system took control of the drone. After all, it was supposed to be following orders from the mission abort system. After a lot of tinkering, trial and error, I noticed that the main system kept sending commands to the drone. Both systems were sending different commands to the drone at the same time.

I read up on the documentation for pynput (pynput 1.4.5, 2019) and found that the keyboard input check was running on another thread in parallel with the rest. The function executed when a key is pressed was also running on this parallel thread.

The solution was, as already described in the chapter "Streaming", to work with flags. This means: When a mission abort is initiated, a flag called "mission_abort" is set to "True". This flag is checked by the main program several times per second. If it is set, i.e. "True", the thread on which the main system is running takes over the execution of the mission abort. This way, the systems do not interfere with each other.

## 8.7 Problem

I programmed the whole Search and Rescue system based on Sphinx. (Sphinx, 2019) . Every little change I made was tested in the simulation. When I was done with the system, it was time to test the Search and Rescue system on the physical drone in the real world. On a cold Sunday afternoon, I went with a friend of mine to a nearby field. By the time we had the drone ready to fly, our fingers were almost frozen.

I started the Search and Rescue system, and everything went smoothly at first. The system was able to connect to the drone, the drone had a GPS signal and the mission area check worked flawlessly. After various preparations, the drone took off successfully. It climbed to the desired altitude and then just stopped, after two minutes I had to initiate the mission abort.

The strange thing is that the drone performed the mission smoothly in the simulation. According to the Sphinx and Olympe documentation, there should be no differences between the simulated and physical drone. That's why I suspected that the firmware of the simulated drone is not the same as the firmware of the physical drone. Since I programmed the system based on the simulation, the Search and Rescue system only runs properly with the simulated drone. The other systems, such as the navigation system or the person detection system still work smoothly. The physical drone simply ignores some commands of the system.

Since I only finished the development a week before the project was due, I didn't have enough time to fix the bug. Fixing the bug would take a lot of time, as I would have to test every change outside on the physical drone. I am somewhat comforted by the realization that the fault is not mine, but "Parrot's". The firmware should be the same on the physical and simulated drone!

# 9 Function test

For the functional test, I check the fulfillment of the basic objectives. The technical objectives are the basis for the basic objectives. For example, if the mission area cannot be defined, how should the mission area be flown over at all?

Since the Search and Rescue system with the physical drone did not work as planned, I distinguished between the simulated and the physical drone for the functional test:

| Function | Simulation | Physical |
|---|---|---|
| Connection to the drone | ✓ | ✓ |
| Person recognition | ✓ | ✓ |
| Autonomous flying | ✓ | ✗ |
| Distinguishing between people seeking help | ✓ | ✓ |
| Notification to operations center | ✓ | ✓ |

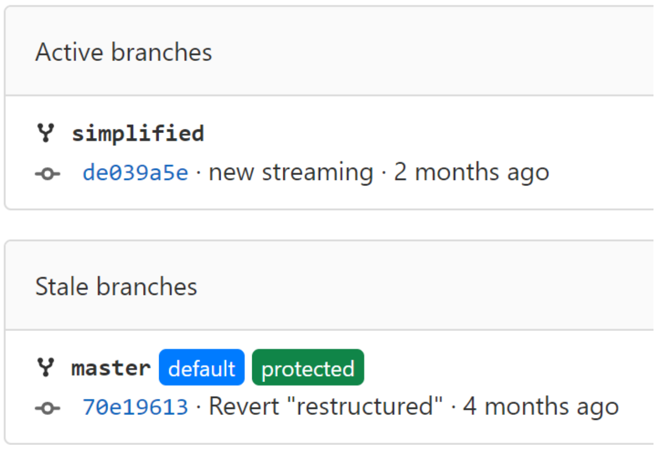
*Table 2: Function test*

# 10 Troubleshooting

Since the Search and Rescue system didn't work as I had hoped on the physical drone, I set out to fix the system after I turned in my project.

## 10.1 Code

I noticed that it was hard to debug the code. On the one hand, I had to test all changes outside with the physical drone, on the other hand, my code was very nested. The original idea was to have one class for all interactions with the drone. On paper this seemed like a very good idea, but in practice it turned out to make debugging enormously more difficult. Such code is called "ravioli code" in technical jargon. Finally, I decided to rewrite a large part of it, in a simplified way! There should no longer be a class that is only there to communicate with the drone. This way I didn't have to always struggle with the interface to the drone. This allowed me to significantly reduce the amount of code, which made debugging easier. To avoid having to rewrite everything, I created a new branch in "Github" and made the changes there:


Active branches

simplified
de039a5e · new streaming · 2 months ago

Stale branches

master default protected
70e19613 · Revert "restructured" · 4 months ago

*Figure 17: Github Branches*

## 10.2 Speed control

While troubleshooting with the physical drone, I noticed that controlling the speed as I had solved it so far was a problem. I simulated the drone's remote joystick and set it to a specific value. This allowed me to limit, but not regulate, the speed of the drone as it searched. Using this technic, the drone accelerates until the forces of the propellers and air resistance balance out. But if a disturbing factor comes along, such as wind, the drone speeds up and slows down again depending on the strength of the wind. This results in bobbing, which makes it difficult for the person detection system to locate the people.

I was able to fix this problem with a regulation loop. Using a regulation loop it's possible to specify a certain speed that the drone should fly. However, I now had the problem that I could only retrieve the current speed of the drone and control its speed using its inclination. It's not possible to tell the drone it should fly a certain speed. Unfortunately, the value of the drone's speed returned was delayed. This made it very difficult to regulate. Fortunately, I came up with a good solution to this. The speed may well be delayed, but the acceleration is not. When the drone is tilted forward, the acceleration changes without delay. This means, I could use the acceleration to control it. If only it was possible to retrieve this data from the drone. I had to find a solution by myself. For this I simply programmed a loop, which stored the current speed of the drone and the current time in a variable at each run. Then I calculated the elapsed time to the last run of the loop and calculated the change in velocity. This gave me the velocity change in a given time, and that is the acceleration. The formula for this looks like this:

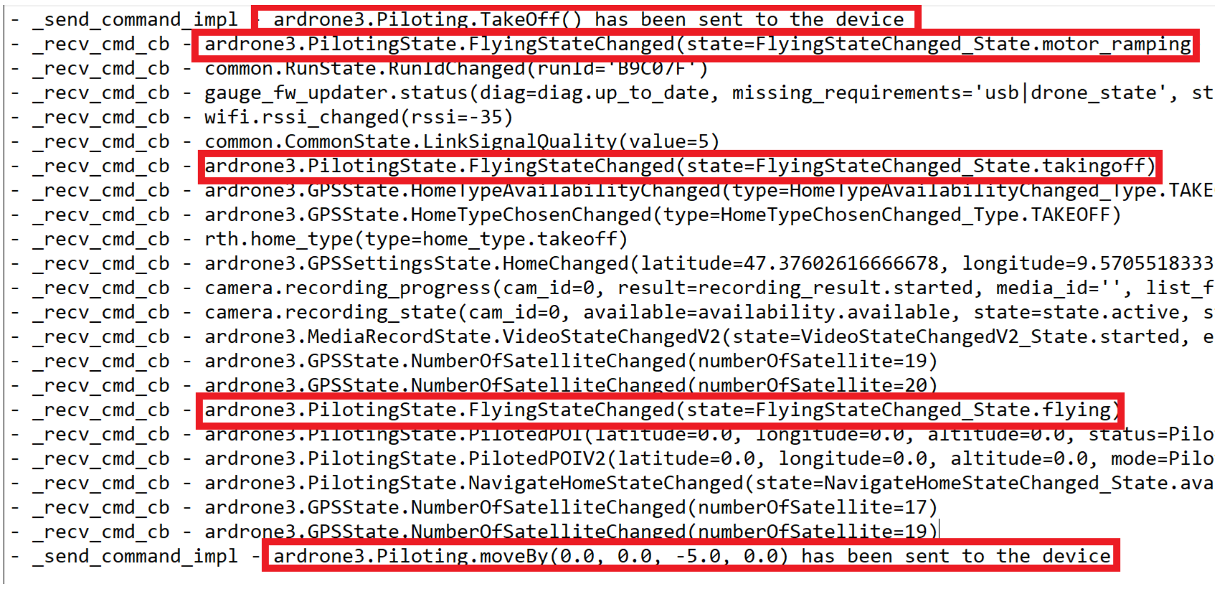
$$a = \frac{\Delta v}{\Delta t} = \frac{v_2 - v_1}{t_1 - t_2}$$

With the acceleration, the speed can now be calculated for a given time. For example, it is possible to determine the speed reached in two seconds. If this calculated speed is now below the target speed, the simulated joystick can be used to accelerate even more. In reality, the joystick would simply be pushed forward. If the calculated speed is now above the target speed, the simulated joystick can be used to reduce the acceleration or even bring it into the negative. This control is performed in a loop several times per second. The formula for this is as follows, with t set to two seconds:

$$V = V_0 + a * t$$

## 10.3 Parrot's failure

While troubleshooting, the drone sometimes worked and sometimes didn't. The drone's behavior was seemingly random. To get to the bottom of the problem in more detail, I put the Search and Rescue software aside and wrote separate small programs. These were to test the individual functions of the physical drone. I even tested some programs written by Parrot themselves on the physical drone, with interesting results. All the little programs worked fine on the simulated drone. But on the physical drone, none of the small programs worked. While it was possible to connect to the drone. The drone was also able to take off on almost all of them, but after that all further commands were ignored. Except the command PCMD, which simulates the joystick. I was also able to determine the reason for it ignoring the other commands. Namely, the problem is because most of the commands of the drone can only be executed when it's in a certain state. For the "MoveTo" or "MoveBy" commands, the drone must be in the "Hovering" state. Otherwise, the commands are simply ignored. This was the case! When the drone is launched, it lifts off the ground and flies up about a meter. This puts it in the "flying" state. But as soon as it finishes flying up, it should switch to the "hovering" state. However, it simply doesn't do that. Even an explicit "hover" command does not put it into the "hovering" state. In the simulation it works fine. Accordingly, the bug seems to be in the firmware of the drone. Unfortunately, I don't have access to this firmware. The error is even visible in the log:



```
- _send_command_impl - ardrone3.Piloting.TakeOff() has been sent to the device
- _recv_cmd_cb - ardrone3.PilotingState.FlyingStateChanged(state=FlyingStateChanged_State.motor_ramping
- _recv_cmd_cb - common.RunState.RunIdChanged(runId='B9C07F')
- _recv_cmd_cb - gauge_fw_updater.status(diag=diag.up_to_date, missing_requirements='usb|drone_state', st
- _recv_cmd_cb - wifi.rssi_changed(rssi=-35)
- _recv_cmd_cb - common.CommonState.LinkSignalQuality(value=5)
- _recv_cmd_cb - ardrone3.PilotingState.FlyingStateChanged(state=FlyingStateChanged_State.takingoff)
- _recv_cmd_cb - ardrone3.GPSState.HomeTypeAvailabilityChanged(type=HomeTypeAvailabilityChanged_Type.TAKE
- _recv_cmd_cb - ardrone3.GPSState.HomeTypeChosenChanged(type=HomeTypeChosenChanged_Type.TAKEOFF)
- _recv_cmd_cb - rth.home_type(type=home_type.takeoff)
- _recv_cmd_cb - ardrone3.GPSSettingsState.HomeChanged(latitude=47.37602616666678, longitude=9.5705518333
- _recv_cmd_cb - camera.recording_progress(cam_id=0, result=recording_result.started, media_id='', list_f
- _recv_cmd_cb - camera.recording_state(cam_id=0, available=availability.available, state=state.active, s
- _recv_cmd_cb - ardrone3.MediaRecordState.VideoStateChangedV2(state=VideoStateChangedV2_State.started, e
- _recv_cmd_cb - ardrone3.GPSState.NumberOfSatelliteChanged(numberOfSatellite=19)
- _recv_cmd_cb - ardrone3.GPSState.NumberOfSatelliteChanged(numberOfSatellite=20)
- _recv_cmd_cb - ardrone3.PilotingState.FlyingStateChanged(state=FlyingStateChanged_State.flying)
- _recv_cmd_cb - ardrone3.PilotingState.PilotedPOI(latitude=0.0, longitude=0.0, altitude=0.0, status=Pilo
- _recv_cmd_cb - ardrone3.PilotingState.PilotedPOIV2(latitude=0.0, longitude=0.0, altitude=0.0, mode=Pilo
- _recv_cmd_cb - ardrone3.PilotingState.NavigateHomeStateChanged(state=NavigateHomeStateChanged_State.ava
- _recv_cmd_cb - ardrone3.GPSState.NumberOfSatelliteChanged(numberOfSatellite=17)
- _recv_cmd_cb - ardrone3.GPSState.NumberOfSatelliteChanged(numberOfSatellite=19)
- _send_command_impl - ardrone3.Piloting.moveBy(0.0, 0.0, -5.0, 0.0) has been sent to the device
```

*Figure 18: Drone log*

1. I'm sending a launch command to the drone.
2. The state of the drone is changing. The engines are accelerating.
3. The state of the drone changes. The drone takes off.
4. The state of the drone changes. The drone flies to about one meter.
5. The state of the drone does not change. The drone simply hovers in the air. (Here the drone should change to the "Hover" state).
6. The MoveBy command is sent to the drone and ignored.

I have contacted Parrot after discovering this error of the drone. It could be that I am doing something wrong. From my point of view, it is unlikely that I did something wrong, because the example programs from Parrot also do not work. Unfortunately, I never got an answer.

# 11 Further development

With the advent of 5G technology, there are new opportunities to further develop the Search and Rescue system. The biggest change I would like to implement would be to run the AI in a datacenter. Normal laptops don't have enough power. This would also allow the AI to be continuously improved - based on the central architecture. That way, the respective users of the system wouldn't have to download updates to always have the latest version of the AI. The whole thing works like this:

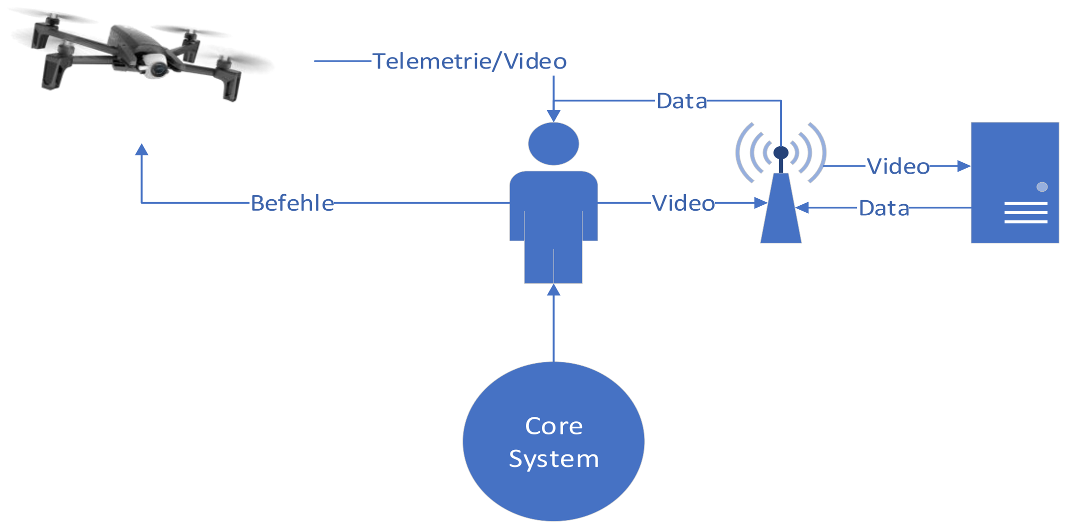


Figure 19: Further development idea

The core system runs on a laptop or tablet. The human interacts with the tablet and provides mission information to the system or monitors the mission. The laptop or tablet sends the commands to the drone and the drone responds with telemetry and video data. This video data can be transmitted to the data center via 5G. The bandwidth of 5G is sufficient for this purpose. At the datacenter, an AI looks at the images and searches for people or checks the pose. If the AI finds anything, that information is transmitted back to the laptop. Of course, 5G doesn't exist everywhere in this world yet, especially not in developing countries. But it is also just a further development idea.

## 12 Reflection

Despite some doubts and nerve-wracking phases, I can happily announce that I have managed to combine the two technologies "Artificial Intelligence (AI)" and drone to program an autonomous Search and Rescue drone. The Search and Rescue system itself works perfectly!

All goals were achieved. A detailed evaluation of the possible drones was carried out. The software was written. I reviewed different products on the market and made a functional video. The video shows how the system works and the process of a mission. Unfortunately, I could not make a video of the physical drone as it did not fly as planned. I also did a functional test to show that the drone works properly in the simulation, the functional test was done for the physical and the simulated drone.

I had a lot of fun with this project. I love to tackle a new, demanding challenge. Without much prior knowledge of Python, Person Recognition, Pose Recognition and Object Based Programming, I blindly jumped into this project. I dealt with the mentioned topics intensively and mastered them successfully.

I'm honestly surprised that the Search and Rescue system basically works. It's just very unfortunate that the physical drone doesn't fly as I would like it to. What's especially frustrating is that the fault is not mine. I should have used the DJI drone instead of the cheaper Parrot. Although this has some disadvantages compared to the Parrot drone, the DJI platform is a lot more robust.

Although this project was a lot of fun, it was also extremely stressful. Every evening since the start of the in-depth work, I have been busy with it and done nothing else. Just acquiring the skills to implement this project in the first place took up a lot of time. I did not count these countless hours towards the actual work. Maybe I should have chosen a simpler project, but then I wouldn't have learned as much. In the end, I think that all those sleepless nights were worth it!

This project was certainly valuable for my future. All the new skills I have acquired during this time will be put to good use in my future. It's not just about knowledge, but also perseverance, patience and tackling problems constructively.

My goal remains to get the Search and Rescue system working on the physical drone as well. Maybe in the future my work can really be used in real Search and Rescue missions to save lives. Because that's what life is ultimately about, and that's my mission, to make the world a better place.

## 13 Bibliography

*An Intro to Threading in Python*. (2019). (Real Python) Retrieved on November 24, 2019, from https://realpython.com/intro-to-python-threading/

*ANAFI*. (2019). (Parrot) Retrieved on November 24, 2019, from https://www.parrot.com/de/drohnen/anafi

*Cascade Classifier*. (2019). (OpenCV) Retrieved on November 24, 2019, from https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

*Cascade Classifier Training*. (2019). (OpenCV) Retrieved on 11/24/2019, from https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html

*COCO 2018 Keypoint Detection Task*. (2018). (COCO) Retrieved on 11/24/2019 from http://cocodataset.org/#keypoints-2018

*These are the drones of the future*. (2019). (Manpower) Retrieved on November 24, 2019, from https://www.manpower.de/neuigkeiten/der-joblog/detail/trends-und-spielereien-das-sind-die-drohnen-der-zukunft-563/

*Deep learning based human pose estimation*. (2018). (Learn OpenCV) Retrieved on November 24, 2019, from https://www.learnopencv.com/deep-learning-based-human-pose-estimation-using-opencv-cpp-python/

*Drone Efficacy Study*. (2018). (Eena) Retrieved on November 24, 2019, from https://eena.org/document/eena-dji-programme-drone-efficacy-study/

*Drone software that saves lives*. (2018). (DroneSAR) Retrieved on November 24, 2019, from https://www.dronesarpilot.com/

*GeoPy documentation*. (2018). (GeoPy) Retrieved on November 24, 2019, from https://geopy.readthedocs.io/en/stable/

*Human Psoe Evaluator Dataset*. (2018). (VGG) Retrieved on 11/24/2019, from http://www.robots.ox.ac.uk/~vgg/data/pose_evaluation/

*JavaScript Object Notation*. (2019). (Wikipedia) Retrieved on November 24, 2019, from https://de.wikipedia.org/wiki/JavaScript_Object_Notation

*Coastal Intelligence*. (2019). (Wikipedia) Retrieved on November 24, 2019, from https://de.wikipedia.org/wiki/K%C3%BCnstliche_Intelligenz

*Matrice 200 Series*. (2019). (DJI) Retrieved on 11/24/2019, from https://www.dji.com/matrice-200-series

*Mavic Pro*. (2019). (DJI) Retrieved on November 24, 2019, from https://www.dji.com/mavic

*MPII Human Pose Dataset*. (2018). (MPI) Retrieved on 11/24/2019 from http://human-pose.mpi-inf.mpg.de/

*Nokia Drone Networks*. (No date). (Nokia) Retrieved on November 24, 2019, from https://www.dac.nokia.com/applications/nokia-drone-networks/

*NumPy*. (2019). (NumPy) Retrieved on November 24, 2019, from https://numpy.org/

*Olypme Documentation*. (2019). (Parrot) Retrieved on November 24, 2019, from https://developer.parrot.com/docs/olympe/

*OpenCV*. (2019). (OpenCV) Retrieved on November 24, 2019, from https://opencv.org/

*Pixhawk*. (2019). (Pixhawk) Retrieved on November 24, 2019, from https://pixhawk.org/

*Pose Estimation*. (2017). (arxiv) Retrieved on November 24, 2019, from https://arxiv.org/pdf/1611.08050.pdf

*pynput 1.4.5* (2019). (pypi) Retrieved on 11/24/2019, from https://pypi.org/project/pynput/

*Python*. (2019). (Python) Retrieved on November 24, 2019, from https://www.python.org/

*Sphinx*. (2019). (Parrot) Retrieved on November 24, 2019, from https://developer.parrot.com/docs/sphinx/whatissphinx.html

*Tello*. (2019). (Ryze) Retrieved on November 24, 2019, from https://www.ryzerobotics.com/tello

*Tello Python*. (2019). (Github) Retrieved on November 24, 2019, from https://github.com/dji-sdk/Tello-Python/tree/master/Tello_Video_With_Pose_Recognition

*TkInter*. (2019). (Python) Retrieved on November 24, 2019, from https://wiki.python.org/moin/TkInter

## 14 Thanks

I would like to thank my family who supported me during these stressful times and let me film them. A special thank goes to my learning partner Lars Schnitzler. He advised and helped me well in many situations. I would also like to thank Joshua Häsler, who actively supported me in testing the system. Another special thank you goes to my father Urs Rinderer; he did the tedious work of correcting my orthography.

## 【評語】190043

This project designs and implements a search and rescue drone. It integrates the auto-pilot technology and computer vision technology. However, the design was only tested in a simulator and the real system cannot yet work in the real world. It would be nice to see this system implemented and tested in the real environment.