How drones controlled by a mobile application can be utilized to find sheep carrying radio transmitters

Linnestad, J. Siggard, Ødegaard, Ole Håkon

May 11, 2020

Contents

Figures

Tables

Code Listings

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs) or drones have become more popular in the last couple of years. It started as a military invention, but has also been used both in the industry and for other special operations. Examples of applications are; search and rescue operations [1], wildlife monitoring [2], and medicine and equipment delivery services [3]. Drone prices have decreased lately, which have led to an increased popularity also among hobbyists and small enterprises.

Throughout the last decades, agriculture has become more industrialized and centralized. It has gone from many small to few but large farms that are capable of producing a more considerable amount of food and other crops. To be able to monitor, manage, and run large areas of crop more efficiently, farmers apply innovative and technological tools – for instance, drones. The use of drones makes the job easier, safer and more time saving.

The ability to monitor farms from a bird's-eye view is beneficial for farmers, especially in rough terrains. Drones can access large areas of ground efficiently with a low amount of risk. Human pilots can control the aircraft along its path but may have difficulties observing the drone's environment. When the drone is out of sight, it is reasonable to assume that the drone itself will be better suited to sense the surrounding environment and act accordingly to preserve a safe and efficient flight. The drone will have to perform its mission automatically, without any interaction with the pilot on the ground.

Chapter 2

Background

Many Norwegian farmers place their livestock in forest or on mountains during the summer. The animals feed on natural grass fields and water resources in the area, while farm pastures are left to grow. This makes animal farming less expensive and time consuming.

On the other hand, as animals are left in the wilderness, chances of accidents and predator attacks increase. Rough terrains and harsh weather conditions can endanger the animals' physical and mental health, especially young animals. To cope with these challenges, farmers, individually or through a cooperation, walk the feeding areas making sure the animals are healthy and unharmed. Such operations often involves long walks over hills and valleys and might be dangerous for the animal keeper. Farmers need a better way to watch over their stock.

One possible solution is to take advantage of a drone's elevated view and use this to monitor the livestock in the terrain. Drones can cover large areas of ground in a short amount of time without endangering farmers or animals. At the same time, the aircraft can be equipped with tools, such as radio receivers and thermal cameras, to aid the animal surveillance.

2.1 Sheep herding on outfield pastures

Animals' health are protected by Norwegian laws and regulations, for instance through the Animal Welfare Act [4] and Regulations of welfare for sheep and goats [5]. All animal farmers are obligated to treat their livestock well and protect them from dangers and unnecessary stress. This also applies for animals on outfield pastures.

2.1.1 Statistics

According to statistics from 2019, more than 2,2 million of the Norwegian livestock graze on outfield pastures during the summer season. [6] A majority of these animals are sheep. At the same time over one hundred wolves are registered as fully or partly living in Norwegian woods. With an average of 248 incidents [7] per year, wolf is just one of many reasons for deceased sheep on pastures. It is estimated that a total of more than 100 000 sheep died on outfield pastures during the 2016 season. [8] More than 65% were assumed deceased from other reasons than predators, such as illness, accidents, or not found during sheep gathering. From these statistics, it is clear that a large number of animal deaths can be avoided. By introducing new tools and technology for better and more efficient animal herding we hopefully can see a decrease in animal deaths on outfield pastures.

2.1.2 Today's situation

To investigate how sheep herding can be improved, we need to understand today's situation, hence procedures and tools in use. According to Linnestad and Ødegaard [9], shepherds do walks on outfield pastures multiple times a week. All of them also says that trips are planned and carried out as a cooperative arrangement between multiple farms. Farmers with animals grazing in the same areas often have some kind of partnership to increase the efficiency of the shepherding. Such arrangements increase the quality of the shepherding, but as animals often are widely spread over large areas of woods and hills, it can go several days before injured or dead animals are discovered and brought home. Sheep tend to hide themselves whenever injured or frightened. Because of this, they are often hard to discover – especially in areas with rough terrain and thick forest.

Linnestad and Ødegaard states that most of the farmers use binoculars when they are watching over their sheep. With binoculars, sheep can be seen from a longer distance, which makes it more efficient and less exhausting for the farmer. At the same time it seems that many are struggling to fulfill the authority's demands on how often one should watch the animals. This is where drones come in handy!

2.2 Environmental considerations

To evaluate the use of drones for animal surveillance we need to consider different environments in which the drone may operate. Such environments may change based on weather conditions, time of the year, time of the day, locality, or other factors. For the drone to be able to fly safely and perform its objective, we present important considerations and evaluate them with respect to the present problem.

2.2.1 Changes in wind and weather conditions

In this paper, we only consider the use of drones in outdoor environments. In an outdoor environment, changes in weather and wind conditions can happen quickly. For small and light drones, heavy wind can potentially cause the drone to lose control and, in the worst case, make it crash, hit an object, or damage the aircraft's components. It is primarily the responsibility of the pilot on the ground to make sure the weather conditions are ideal for flying the aircraft. Strong wind or heavy rain might suddenly appear and damage the drone, so the pilot must take the necessary precautions before taking off.

Studies [10] have shown that when the wind is heading in the opposite direction as the flight direction, the drone will use less power. The battery consumption will be lower when flying against the wind because of the increased lift the drone gets when it is moving horizontally. When the drone hovers in the same position, it does not have any lift. This means that the propellers have to do all the work. As more air flows against the flight direction, less power is required to keep its height.

2.2.2 Stationary objects

There are many kinds of stationary objects one has to consider when flying a drone outside. Some examples are houses, builds, poles, antennas, towers, and power lines. The drone should keep a safe distance away from such objects to avoid a crash.

Many drones use a built-in compass to pinpoint the location and movement of the aircraft. High voltage power lines produce electromagnetic fields. If magnetic fields interfere with the drone's magnetic compass, it can affect the drone's maneuverability. Depending on the distance to the power line, the results can be catastrophic. Radio towers can also interfere with the signals moving to and from the aircraft, including radio signals from the remote controller and GPS signals to satellites.

Outdoor environments often have vast vegetation. Trees and bushes with all kinds of shapes might appear in the drone's flight path. If this is the case, the drone needs to take action to avoid crashing into the trees. If a rotor from the drone hits a branch, this can be catastrophic for the aircraft. It will not just damage the rotor, but a free fall from a considerable height down to the ground can also damage other components of the aircraft.

Many drones are equipped with obstacle avoidance systems. Such systems use sensors to measure the distance between the aircraft and the surrounding obstacles. If the distance gets too small, the aircraft's flight controller will take necessary actions to avoid a crash. How well these systems work and how accurate they are might vary based on the drone.

2.2.3 Moving objects

Drones should fly at a safe distance from other flying objects, like birds, other drones, and crewed helicopters and aircraft. Hitting other flying objects can cause a hazard, not only to the objects themselves but also to any personnel, both on the ground and in the air.

There have been multiple incidents [11, 12] where birds, and especially eagles, have attacked flying drones. One assumption is that bird attacks are more likely to happen if the drone is flying close to a bird's nest. To avoid such incidents, one should avoid flying near nesting areas.

2.2.4 Regulations

Regulations on the use of drones vary depending on the country. Since this is a Norwegian study, we focus on laws and regulations conducted by the Norwegian government and other aviation authorities.

The Norwegian Civil Aviation Authority (CAA) [13] distinguishes between *commercial use of drones* and *drones for leisure*. A commercial user is a person or organization making money by flying a drone. If the drone is a hobby or sport, it is defined as a leisure activity. Different rules and regulations may apply to the user depending on several parameters, like the drone type, the size of the drone, the weight of the drone, and the purpose of flying the drone.

Commercial drone operators are required to apply for a flight certificate. The CAA operates with three different certificates. Each of them has different restrictions and requires different applications (see Table 2.1).

Type of certificate	Restrictions	Application
RO1	• Max weight: 2.5 kg • Max speed: 60 knots • Requires a visual line of sight to the drone • Restricted to flight in daylight	Letter to the CAA with information about the organization and drone type
RO2	• Max weight: 25 kg • Max speed: 80 knots • Requires a visual or extended visual line of sight to the drone	• Pass online exam • Receive approval on an application containing a risk analysis and an operation manual for the use of the drone
RO3		• Pass online exam • Receive approval on an application containing a risk analysis and an operation manual for the use of the drone

Table 2.1: Comparison between the three different operator classifications. RO = RPAS (Remotely Piloted Aircraft Systems) Operator

Top Five Rules

The CAA has a list on their web page containing the top 5 most important rules for flying a drone as a hobby. These rules are developed based on the Norwegian regulation *Forskrift om luftfartøy som ikke har fører om bord mv.* [14] and are considered as good guidance on how to operate a drone. The top five rules are:

1. The drone should always be kept within your sight and operated in a mindful and considerate manner. Never fly near accident sites.
2. Never fly closer than 5 km from airports unless you have explicit clearance to do so.
3. Never fly higher than 120 meters off the ground.
4. Never fly over festivals, military facilities or sporting events. Keep a distance of 150 meters.
5. Be considerate of others privacy. Take note of the rules concerning photos and films of other people.

The first rule states that the drone should be kept within the drone operator's sight. Since the goal of this project is to test our hypotheses the drone will be kept within sight during all tests. Rules 2, 3, and 4 are fulfilled by performing the tests on large grasslands, football fields, or in the woods. Any photos or videos captured by the drone during any of the tests are handled confidential and deleted after each test. No photographic material containing personal information is stored or shared on public channels.

Privacy

Most drones are equipped with a camera device and other sensor components. If any of these components capture or record information, e.g., images, sound recordings, or videos, about a person, privacy regulations apply. The Norwegian Data Protection Authority (DPA) has developed five recommendations about data privacy and drones:

- Minimize the amount of recorded information about humans and other identifiable objects
- Make yourself visible when operating the drone
- Do not use pictures or information captured by the drone for other purposes than originally intended
- Make sure all information captured by the drone is safely stored. Evaluate the security mechanisms in the software on the drone and any device storing the information.
- In commercial operations, make sure to have a valid agreement with clients regarding the ownership of the captured information.

2.3 Drones

As previously mentioned drones are unmanned aerial vehicles. This description fills a large range of air crafts. This section will go through common types of drones, compare them, and go in detail of the most common consumer drone, namely the quad-copter, specifically how it works.

2.3.1 Different drone designs

There are several types of drones. The definition may include several things from weather-balloons to drones used in military operations such as the MQ–9 Reaper with a wingspan of 66 meters. [15] As this paper focuses on covering an area, specifically wilderness where farm animals are herded, it makes sense to look at common types of drones used in the consumer market. The following are some common drone types [16]:

- **Multi Rotor Drones** - these drones have multiple rotors, most common is the quad-copter having four rotors. They are relatively cheap to make. They can take off from a standstill and hover. It is not efficient as it has to constantly use energy to stay in the air.
- **Fixed Wing Drones** - these are more like normal airplanes with a wing and a propulsion system. They need a runway or a launcher to start flying and are not able to hover, they are however much more efficient not having to hover meaning energy is used to move it forwards rather than up.
- **Single Rotor Drones** - these are more akin to helicopters with a single rotor blade on the top, and usually on at the tail spinning vertically to keep it in balance. They are harder to manufacture.
- **Hybrid VTOL** - these are hybrids of fixed wing and rotor based designs and inherit the advantages of taking off and landing anywhere, while being energy efficient in the air.

They all have different capabilities and are used for different purposes. Due to the drone used in this paper being a quad-copter, it is the drone design that will be highlighted in the next section

2.3.2 Quad-copters

Quad copters are designed with four propellers. In helicopters there are a single rotor where each blade on the propellers can tilt around itself and this is used to stabilize and move the aircraft. In quad-copters however, the blades are rigid, meaning movement and control come from the difference between the four propellers in uplift. By spinning a propeller slower, the aircraft will tilt and move in that direction. To stay stable a quad-copter moves each propeller every other direction. This is illustrated in figure 2.1. If all propellers were spinning in the same direction the drone itself would spin in the same direction. By having each rotor moving in the opposite direction the forces making the drone turn are nullified.

A quad-copter moves by changing its pitch, roll, and yaw as well as throttle. The throttle is simply how fast the propellers spin. If the aircraft is hovering, a decrease in throttle will cause the aircraft to descend. If the throttle increase it will ascend. The following list shows how the aircraft acts to changes in pitch, roll, and yaw:

- *Pitch* makes the drone move forward and backward. The aircraft will tilt either forward or backward and moves in the direction it tilts. To pitch the

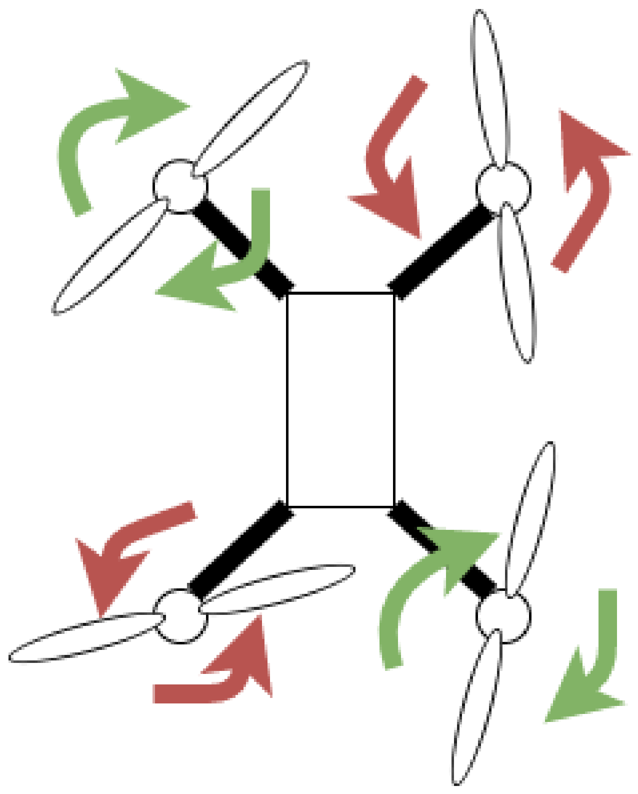


Figure 2.1: Figure illustrating alternative direction of propellers on quad copters. Green is clockwise, red is counter-clockwise

the aircraft forward, the rear propellers increase their thrust.

- *Roll* is a rotation around the X-axis. The aircraft will tilt either to the right or to the left and will move in the tilted direction. To pitch the aircraft to the right the two left propellers increase their thrust.
- *Yaw* is a rotation around the Z-axis. A change in yaw makes the aircraft direction (or heading) change. An example: the aircraft is hovering and face north. It yaws clockwise and stops when it has turned 90 degrees. It is now facing east. The yaw is performed by increasing the throttle of propellers spinning in the same direction.

Chapter 3

The drone

Given the difference in drones, both in capabilities, customizability and availability it is important to look at the drone used for tests and in the overall system. All though a lot of the system is built to be general, some parts of the system depend on the drone itself. This chapter gives an overview of the drone used in this paper. It first given a introduction of the manufacturer, followed by a look at the specifications and discuss how this affect the system developed in this paper.

DJI is the largest consumer-based drone manufacturer in the world. The Shenzen based company grossed over $2.7 billion in 2017.[17] Drones produced by DJI are used in everything from photography and film-making to surveillance and agriculture. They offer a large variety of drones for different purposes, both for general consumers, as well as enterprise drones with specialized purposes. The different drone series they offer are as follows:

- Mavic - Foldable all-purpose drones
- Spark - Small relatively cheap drones
- Phantom - Drones with advanced aerial mapping capabilities
- Matrice - Highly durable and configurable, increased load capability

There are also several variations within each series. An example would be the Mavic series where the smallest drone Mavic Mini has a takeoff weight of 249 grams versus the Mavic 2 Enterprise having a max takeoff weight of 899 grams.

3.1 Drone specifications

In this paper, a DJI Mavic 2 Enterprise Dual is used to test automatic flight. The Mavic 2 Enterprise Dual is an all-purpose foldable drone, with both a regular camera and thermal vision. It has four electric motors, a lithium rechargeable battery, as well as obstacle detection in all directions. The next section highlights features of the drone relevant to the system in this paper, both in terms of hardware and software. All specifications are from the official user manual. [18] Table 3.1 shows some specifications relevant to this project.

Spec	**Value**
Takeoff weight	899 grams
Max takeoff weight	1100 grams
Dimensions(Length/Width/Height in cm	32.2/24.2/8.4
Max accent speed	5 m/s in S-mode 4 m/s in P-mode
Max descent speed	3 m/s
Max speed	70km/h in S-mode 52km/h in P-mode
Max flight time	31 minutes at a consistent 25km/h
Operating temperature	-10°C to 40°C

Table 3.1: Some relevant specifications of the DJI Mavic 2 Enterprise Dual

3.1.1 Modes of flying

The DJI Mavic 2 Enterprise Dual offers three different modes of flying. The modes change how the aircraft responds, both in terms of speed and features such as obstacle avoidance. The different modes are as follows:

- *P-mode* - the default mode. It works best when the GPS is strong and limits the maximum flight speed. It enables obstacle detection and avoidance. This mode must be enabled to control the drone pro grammatically.
- *S-mode* - sport mode. This mode disables obstacle sensing and has a much higher maximum flight speed. It gives the greatest direct control of the drone
- *T-mode* - tripod mode. Similar to P-mode, but with much slower maximum speeds. Used for stable shooting of images.

As this paper aims at automatic flight efficiently P-mode is the obvious choice.

3.1.2 Electric motors and propellers

The combined maximum takeoff weight of all four motors combined is 1100 grams. As the drone itself weigh 899 grams this allows for additional equipment of upwards of 201 grams. An example would be adding a radio transmitter for use in sending and receiving signals from the ground a relaying them to a user. The ascent speed is a maximum of 5 m/s and the descent speed a maximum of 3 m/s. This means that from hovering a ground-level to a maximum height of 120 meters the drone would use 24 seconds, and 40 seconds back down. This is actually a significant time as the maximum flight time of the drone is roughly 31 minutes. Just ascending and descending would take ∼3% of the total time of one flight with a fully charged battery.

The propellers featured on the Mavic 2 Enterprise are detachable. Each is connected to a motor with specific color-coding to indicate which motor it is supposed to be attached.

3.1.3 Battery

The battery comes as a detachable battery pack. This means one can have multiple batteries and easily switch when one runs out. As stated in the previous section the battery pack last for approximately 31 minutes. This at a moderate speed of 24 km/h. Using these number the drone should be able to fly around 12 km given it flies in a straight line at the same altitude under optimal weather conditions.

3.1.4 Remote controller

The DJI aircraft is controlled remotely by a physical controller. This controller includes two sticks for controlling the aircraft, buttons to issue commands such as to start filming or take a picture, as well as a screen used to show the state of the aircraft, such as possible errors, collision, or altitude. There are two antennas attached to the controller. The transmission distance is default to 5000 meters unobstructed. The controller can be connected to a smartphone to issue commands from the phone, change settings and see a video feed from the camera on the drone. The connected phone can communicate with the drone through either the official DJI app or a custom-made one. Figure 3.1 shows an overview of how the different hardware components communicate.

3.1.5 Object detection and avoidance

The DJI Mavic 2 Enterprise also has a vision system consisting on sensors around the drone. Using the vision system the aircraft can detect obstacles in its path. Depending on the mode of the aircraft, it will try to avoid the obstacle. Either by stopping, flying over, or around, depending on the settings defined for the drone. This is built in, and will not have to be implemented.

3.1.6 SDK

DJI offers Software Development Kits (SDK) to create custom functionality and applications to the system. The SDK provides classes and functions so that one can programmatically access the built-in functions of the drone. The different official SDKs provided by DJI are as follows:

- Mobile SDK - includes SDKs for both iOS and Android
- UX SDK - includes UX drop-in components to integrate functionality swiftly
- Onboard SDK - Used to add functionality directly to the aircraft
- Payload SDK - Allows for third-party hardware to communicate with the aircraft
- Windows SDK - Allows Microsoft Windows applications to access the aircraft

The SDKs have access to several parts of the drone. An example is the iOS SDK having access to the following:

- High and low-level flight control

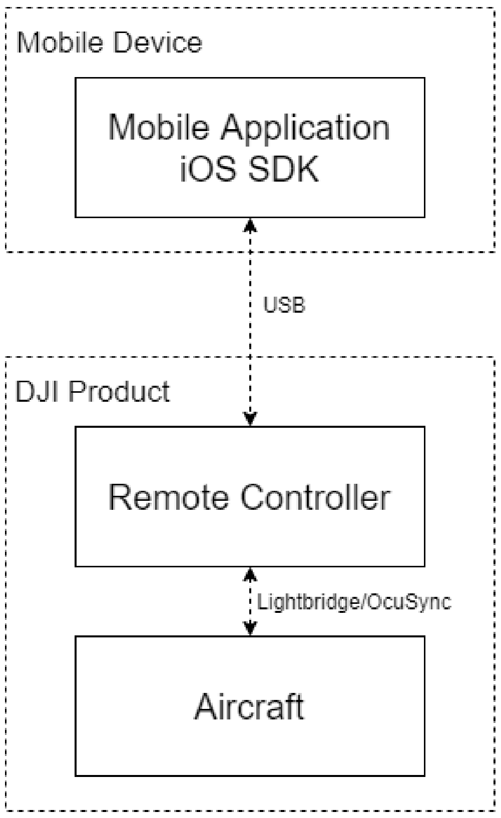


Figure 3.1: Overview of the communication between the mobile application using the iOS SDK, the remote controller and the drone.

- Aircraft state through telemetry and sensor data
- Obstacle avoidance
- Camera and gimbal control
- Live video feed
- Access to drone media files
- Route planning for automatic flight
- State information from the battery and remote controller

Of these, the route planning, flight control, obstacle avoidance, and state information is of high relevance to the project. Figure 3.1 shows how the SDK is used in an application to communicate with the aircraft through the remote controller. The use of the SDK in this project is described in greater detail in chapter 4.

Chapter 4

Automatic Flight

4.1 Introduction

This chapter describes the choices and implementation of automatic flight of the DJI Mavic 2 Enterprise. It is divided into sections from initial research to implementation details, automatic flight testing, results, discussion, and a conclusion. The goal behind the test in this chapter is to verify the possibility of automatic flight along a defined path, as well as gather metrics such as completion time of different tasks. The main question answered in this section is: "Can you fly a DJI Mavic 2 Enterprise in a predefined pattern automatically?". The results provide a foundation for the next parts of the system, such as area path planning and user-defined areas.

4.2 DJI SDK

As the drone provided by the institute for use in this thesis is a DJI Mavic Enterprise Dual, some considerations must be made due to the limitations of custom functionality. As described in chapter 3, DJI offers SDKs to control the drone programmatically, and receive feedback from sensors. This section looks at the different SDKs provided by DJI and highlights the pros and cons in relation to this project. The features of the different SDKs are shown in table 4.1.

4.2.1 Mobile SDK

The mobile SDK communicates with the aircraft through a mobile application connected to the remote controller. The mobile SDK is released both for iOS and Android. Combined iOS and Android hold 100% of the market share in the world with Android at ~87% and iOS at ~13%. [19] In Norway the two operating systems are more evenly shared at ~55% for iOS and ~45% for Android. The iOS SDK is available in the programming languages Objective-C and Swift, while the Android SDK is available in Java.

Pros

The scope of this project includes a user interface with a map to define the area to be covered. A mobile application gives the apparent advantage of having both a touch screen to both display information and to register input from the user. Medienorge says that 95% of all Norwegians have access to a smartphone; this means a system built for smartphones is highly available to the public. [20] In addition to the high availability, a smartphone is usually compact in size and easy to transport. Other hardware such as laptops a more cumbersome to use in remote places such as in terrains like woods or mountains due to their size and battery life. The Mobile SDK also includes a lot of functionality relevant to this project, such as missions, battery sensory data, and object detection. Missions are ways to automate the drone, and this includes Hotspot Missions, where the drone will circle a spot, Follow-Me Missions where the drone follows an object, and Waypoint Missions where missions are defined in the forms of waypoints the drone will fly to.

Cons

As can be seen in table 4.1 the mobile SDK lacks a lot of the features of the Onboard SDK, especially with regards to missions, as the Onboard SDK currently has support for a new mission system that is in beta REF. Another limitation is the need for a smart phone to use the system. You get three hardware parts, the phone, the controller and the drone itself. This means the total cost of the system increases, however as most people have a smartphone this is not a practical limitation.

4.2.2 UX SDK

The UX SDK is a collection of already finished UI elements using the Mobile SDK; as such, the UX SDK has a subset of the features of the Mobile SDK.

Pros

The UX SDK allows for rapid development of standard functionality. The components in the UX SDK are mostly "drop-in," and you get both UI elements such as maps, buttons, and video feed. The components also include all abstracts all the underlying Mobile SDK calls and, as such, work on a higher abstraction level.

Cons

As the components in the UX SDK are already built, they lack customizability. Adding additional functionality to a component is also limited. As the UX SDK builds upon the Mobile SDK, all the cons of the Mobile SDK also apply to the UX SDK.

4.2.3 Onboard SDK

The Onboard SDK is a set of different libraries to be used onboard the DJI drone, and this means the code runs directly on the drone (or controller) as opposed to on an application making calls through the controller using another of the SDKs.

Pros

Table 4.1 shows that the Onboard SDK has the most built-in features of the SDKs. It can do all the same things as both the mobile- and windows SDKs while also giving access to advanced features such as more defined mission and flight controls.

Cons

Due to being able to communicate with the Mobile SDK there are almost no obvious disadvantages to the Onboard SDK. However, when looking at the ease of development, the Onboard SDK uses low-level languages and requires much more in terms of setup. The documentation is also lacking in terms of examples.

4.2.4 Payload SDK

The payload SDK is used to program communication between third-party hardware and the drone.

4.2.5 Windows SDK

The Windows SDK offers almost all the features of the Mobile SDK, but runs exclusively on a Windows Operating System. As such, it has a clear disadvantage due to the most common Windows systems being laptops and thus being less portable than a smartphone.

4.2.6 Conclusion

The project in this paper was done using the Mobile SDK, specifically the iOS SDK. There are several reasons for this choice. Firstly as this system includes both a user interface and automatic flight, the choice of using the Mobile SDK means everything can be packaged into one application. If additional features provided by the Onboard SDK is needed, this can be added by using the Onbaord SDK to Mobile SDK communication feature. Secondly, both authors are in possession of iOS devices, and for convenience sake iOS was chosen over Android.

4.3 Implementation

This section shows an overview of the first iteration of the application in this party.

Feature	**Windows SDK**	**Mobile SDK**	**Onboard SDK**
High and low level flight control	✓	✓	✓
Aircraft state data	✓	✓	✓
Obstacle avoidance	✓	✓	✓
Camera and gimbal control	✓	✓	✓
Live video feed	✓	✓	✓
Pre-defined missions	✓	✓	✓
State and control of battery and remote controller	✓	✓	✓
Access to media stored on camera		✓	✓
Stereo vision video feed			✓
Real-time disparity map			✓
Custom local-frame navigation			✓
Mobile SDK communication			✓
Multi-function I/O ports			✓
Synchronization with flight controller			✓

Table 4.1: Features of the different DJI SDKs

4.3.1 Mobile SDK details

Flight Control

Flight Control is a set of functions and classes related to controlling the aircraft. The SDK does not support a lot of the functionality in flight control. It is documented, but when implemented returns the following error message: "This feature is not supported by the SDK". Therefore these features will not be utilized.

The Body Coordinate System is relative to the aircraft itself. The origin is the center of mass, and it uses three perpendicular axes. X is forward from the aircraft, Y is through to the right, and Z is through the bottom. The rotation of the aircraft uses these same coordinates. *Roll*, *Pitch* and *Yaw* are rotations around the axes of X, Y and Z respectively. the aircraft's orientation is known as its *attitude* and is defined by its rotation around the pitch, roll, and yaw axes.

The World Coordinate System is the coordinates of everything, including the aircraft. The Mobile SDK uses *North-East-Down* system, where X is positive towards the north, Y is positive towards east and Z is positive towards the ground and the origin is an arbitrary location on the earth surface .[21] In the case of the DJI aircraft, it is the point of take-off.

It is important to point out that the built-in flight controller balances the spin of each propeller and that the SDK provides APIs to change the throttle, yaw, roll, and pitch smoothly.

Sensor Data

The SDK provides several different data about the aircraft. Under are some of the data provided by the SDK relevant to the system in this paper.

- *Flight Control* gives access to data such as altitude relative to start, GPS location, the attitude of the drone, and the speed at which it moves.
- *Smart battery* gives access to several different values such as cycles, remaining energy, voltage, and current. It also provides parameters regarding the number of charging cycles, full charge, and remaining life.

Missions

Missions are used to automate flight. Missions are either managed by the application or uploaded and managed by the aircraft. Different types of predefined missions are offered in the SDK. The following are predefined mission types:

- *Waypoint Missions* uses a series of coordinates (waypoints) and automatically fly between each waypoint and can execute an action at each. It is uploaded and executed by the aircraft.
- *Hotpoint Missions* are a fixed point where the aircraft will fly around at a constant radius. Parameters that can be adjusted include speed, altitude, and direction.

- *Follow Me Missions* the aircraft will follow a series of GPS coordinates at a fixed distance.
- *ActiveTrack Missions* uses the aircraft's vision system to track a subject. The user defines the subject with a rectangle and confirms the subject to be followed
- *TapFly Missions* allows the aircraft to move to an object the user clicks on the screen. It uses the visual system to determine the location and flies towards it.
- *Panorama Mission* turns the camera around while taking pictures; these are then combined to create a panorama picture. This feature is, however, not supported by the Mavic 2 Enterprise.

All these mission types inherit from the MissionControl class. The MissionControl class can be used to create custom missions. Missions are created by adding *Actions* to a *Timeline*. An *Action* is operation for the aircraft to perform. A *Timeline* is a series of *Actions* to be performed sequentially. The timeline is uploaded to the aircraft, and the aircraft performs the first action when the timeline is started. Upon completion of an action, the aircraft performs the next. There are several types of actions. Some relevant to this system are listed here:

- *GoToAction* - defines a location and altitude for the aircraft to move to. It also allows the flight speed to be set.
- *TakeOffAction* - makes the aircraft start its propellers and take off from the ground.
- *Landaction* - makes the aircraft land and turn of its propellers.
- *GoHomeAction* - the aircraft will fly to it's home location. The default home location is the point of take-off.
- *ShootPhotoAction* - the camera will take a photo
- *GimbalAttitudeAction* - the gimbal will change direction. This is used to move the camera.
- *AircraftYawAction* - changes the direction of the aircraft.

Together these actions can be used to automate a flight from taking-off, to landing. While in the air, the ShootPhotoAction makes it possible to take a picture automatically, this combined with GoToAction can be used to survey an area automatically.

4.3.2 Implementation of iOS application - first iteration

As the goal behind the first iteration was to get the drone to fly automatically in predefined patterns, as well as to know the capabilities of the drone, the first implementation focus on a small part of the system namely automatic flight in predefined paths. Thus two parts were implemented, connecting to the drone and custom missions.

Connecting to the drone

To use any of the features of the SDK, one first has to connect to the drone. This requires the phone to be connected to the remote controller. When the phone is physically connected to the remote controller, the app has to be registered using the DJISDKManager. The following is a simplified version of registering the app with the SDK:

```
//Example of registration app with the SDK

import DJISDK

class DJI: NSObject, DJISDKManagerDelegate {
   override init() {
      super.init()
      self.registerWithSDK()
   }

   func registerWithSDK() {
      DJISDKManager.registerApp(with: self)
   }

   // MARK: DJISDKManager Delegates

   func productConnected(_ product: DJIBaseProduct?) {
      NSLog("Product connected")
   }
}
```

This code snippet is strapped of error-handling, such as not being connected to the drone and other failures. The SDK manages most of the registration, leaving error handling and overhead control to the developer. In this case the application invokes *registerApp* which upon completion calls *productConnected*. When *productConnected* has been called, features of the SDK that sends commands, or are used to receive sensor data, are available.

Mission

Once the application has been registered with the SDK and the product is connected, missions are available. Missions are explained in section 4.3.1. In the implementation of the first iteration, custom missions were used. The following code snippet shows a very simple use of custom missions:

```
func addTimelineElements() {
   let missionControl = DJISDKManager.missionControl()
   missionControl?.scheduleElement(DJITakeOffAction())
4     missionControl?.scheduleElement(DJIGoToAction(coordinate:
          CLLocationCoordinate2D(latitude: 63.418337, longitude: 10.402769),
          altitude: 5)!)
5     missionControl?.scheduleElement(DJILandAction())
6  }
7
8  func startTimeline() {
9     DJISDKManager.missionControl()?.startTimeline()
10 }
```

The *addTimeLineElements*-function adds actions to the timeline. The following actions are scheduled between line 3 and line 5:

- DJITakeOffAction - returns an action for taking of the aircraft
- DJIGoToAction - returns an action for moving to a certain GPS-coordinate with a defined altitude
- DJILandAction - returns an action for landing the aircraft

When *addTimeLineElements* is called all these actions are uploaded to the aircraft. The aircraft then awaits the timeline to start. This is done by calling the *startTimeLine*-function. Calling the function starts the execution of the timeline on the aircraft. By using custom missions, and these actions, all tests defined in 4.4 may be performed.

4.4 Test plan

4.4.1 Introduction

This section describes the test plan for the first iteration. The goal of the test is for the drone to fly in predefined patterns and upon completion of land automatically. To verify the implementation of the iOS application, several tests will be performed both to confirm that the current implementation works and satisfy all the requirements, as well as the speed at which the drone can complete different actions. The test is mainly to get an idea of the capabilities of the drone, which will be used as a base for further development. For every test, the time to complete each action, and the time between each action will be measured. This will be used to determine the cost of different types of actions, as well as the cost of having multiple actions.

4.4.2 The tests

This section shows all tests as well as why this will be tested. Figure 4.7 shows the planned tests #3-5 with a birds-eye view. Test #1 and #2 are excluded due to not changing coordinates, only altitude.

Test #1

Test #1 is simply taking off and landing. The aircraft should take off and fly to approximately 70 cm, followed by landing safely in the same spot. This test shows that the aircraft can land and take off automatically. These are essential features of the system. Figure 4.1 shows for each step what will be measured.

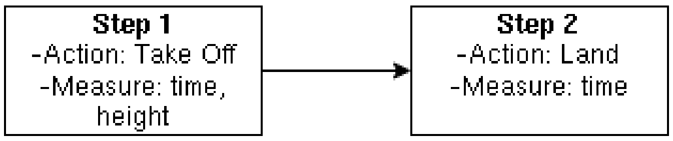


Figure 4.1: Steps in Test #1

Test #2

Test #2 is taking off, increasing altitude to 2 meters and landing. This test shows if the land action can be performed at a higher altitude than where the take-off action finishes. Figure 4.2 shows each step and what will be measured.

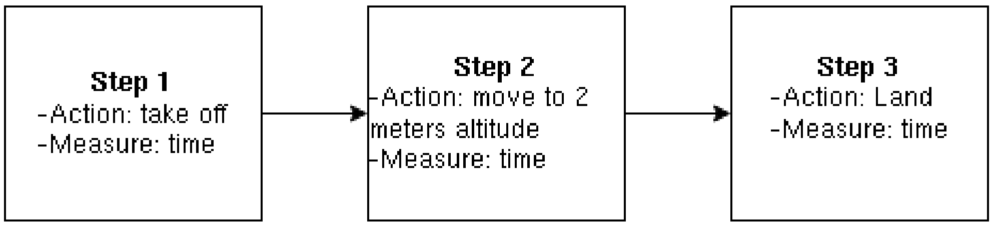


Figure 4.2: Steps in Test #2

Test #3

Test #3 is taking off, increasing altitude to 5 meters, move to a defined location approximately 40 meters from the starting point, return to the take-off location and land. This test tells if the drone can move to a defined location and how fast it moves, it will also show how fast the drone turns 180°. Figure 4.3 shows each step and what will be measured.

Test #4

Test #4 is taking off, move to a location while increasing altitude to 20 meters, return to the take-off location while decreasing altitude to 1 meter and land. This test is performed to check how the drone moves between two points where the altitude is different. Figure 4.4 shows each step and what will be measured.

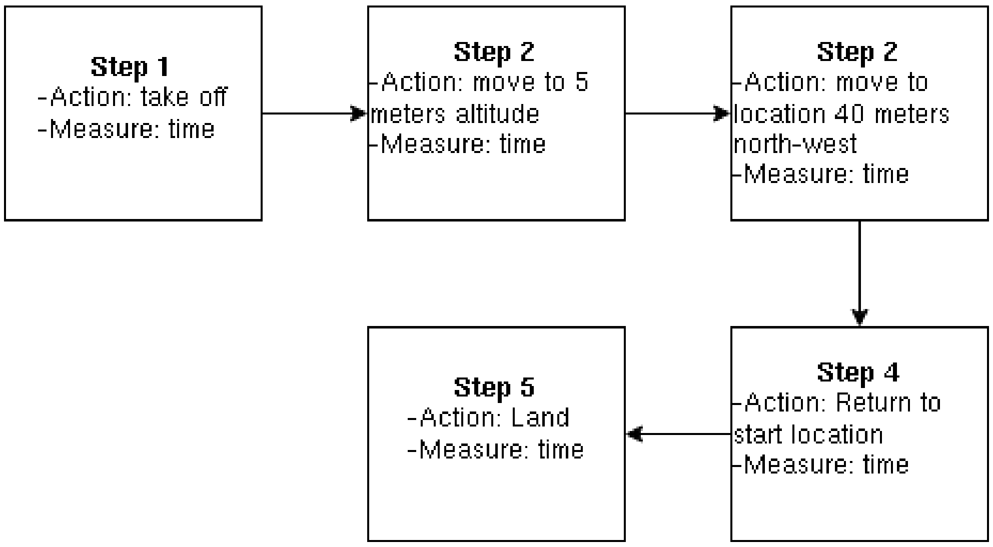


Figure 4.3: Steps in Test #3

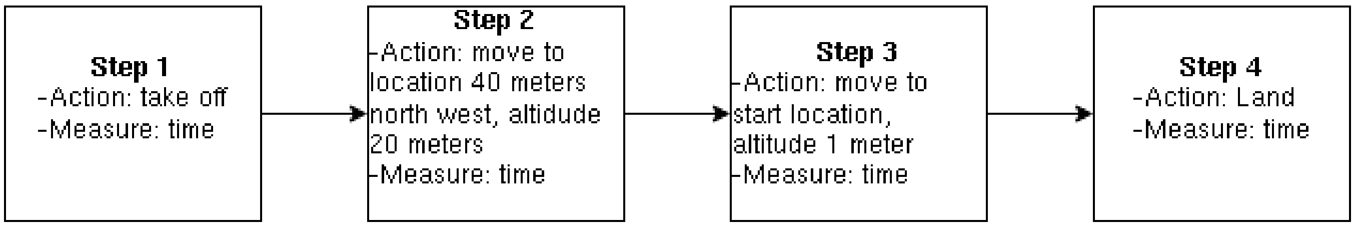

Step 1
-Action: take off
-Measure: time
Step 2
-Action: move to location 40 meters north west, altidude 20 meters
-Measure: time
Step 3
-Action: move to start location, altitude 1 meter
-Measure: time
Step 4
-Action: Land
-Measure: time

Figure 4.4: Steps in Test #4

Test #5

Test #5 is taking off, move to an altitude of 10 meters, move 20 meters, move another 20 meters in the same direction, and return to start point and land. This test is performed to see how much delay there are between consecutive actions. Figure 4.5 shows each step and what will be measured.

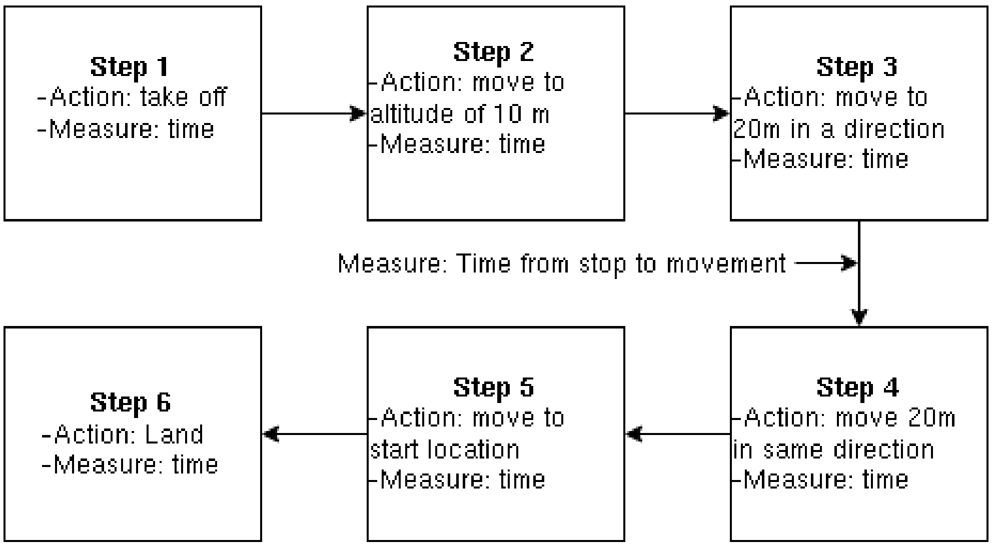

Step 1
-Action: take off
-Measure: time
Step 2
-Action: move to altitude of 10 m
-Measure: time
Step 3
-Action: move to 20m in a direction
-Measure: time
Measure: Time from stop to movement
Step 4
-Action: move 20m in same direction
-Measure: time
Step 5
-Action: move to start location
-Measure: time
Step 6
-Action: Land
-Measure: time

Figure 4.5: Steps in Test #5

Test #6

Test #6 is taking off, and move in a square roughly ten by ten meters at an altitude of 5 meters and return to take-off location and land. This test verifies if the drone can move between several defined locations and return to the starting point. Figure 4.6 shows each step and what will be measured.

4.5 Results

The test was performed on the 10th of March at Dødens Dal in Trondheim, Norway. The time of the test was between roughly 14:30-14:40, and the wind in Trondheim at that time was between 4.3 m/s and 7.6 m/s [22]. This section shows Table REF shows the total time for each test.

Test #	Time (s)
1	16.93
2	26.67
3	60.75
4	55.20
5	74.42
6	95.29

Table 4.2: Time from starting take off to landing for each of the 6 tests

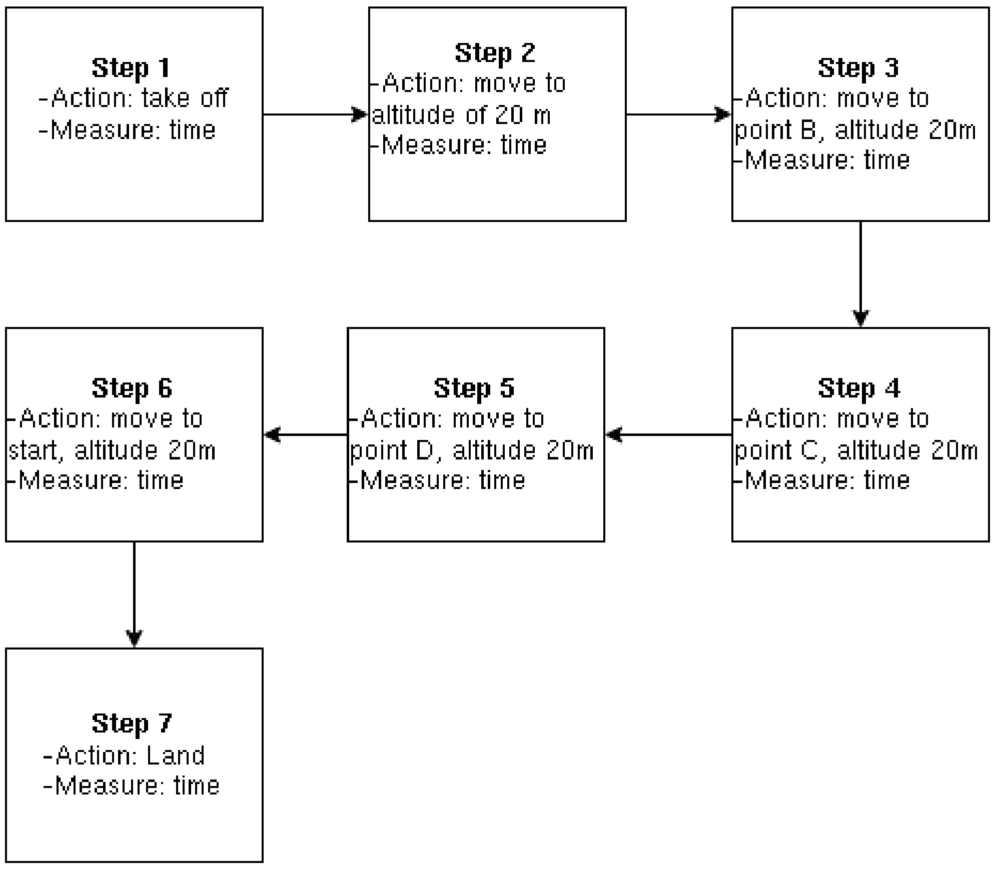

Step 1
-Action: take off
-Measure: time
Step 2
-Action: move to altitude of 20 m
-Measure: time
Step 3
-Action: move to point B, altitude 20m
-Measure: time
Step 4
-Action: move to point C, altitude 20m
-Measure: time
Step 5
-Action: move to point D, altitude 20m
-Measure: time
Step 6
-Action: move to start, altitude 20m
-Measure: time
Step 7
-Action: Land
-Measure: time

Figure 4.6: Steps in Test #6

4.5.1 Take off

As every test includes a take-off step, we can find out the average time to take off. The tests indicate that the drone hovers at an altitude of 1 meter at the completion of the take-off step. The average time is $\tilde{4}.93$ seconds to take off.

4.5.2 Intermediate steps

Between two actions is an intermediate step where the drone hovers. This step averages to 5.39. There is a difference depending on the preceding action. Table 4.3 shows different averages of time in intermediate steps.

4.5.3 Relevant results of each test

Test #1

The total time of the test was 15.93 seconds. Of this time, 7.06 of it was the intermediate step between taking off and landing.

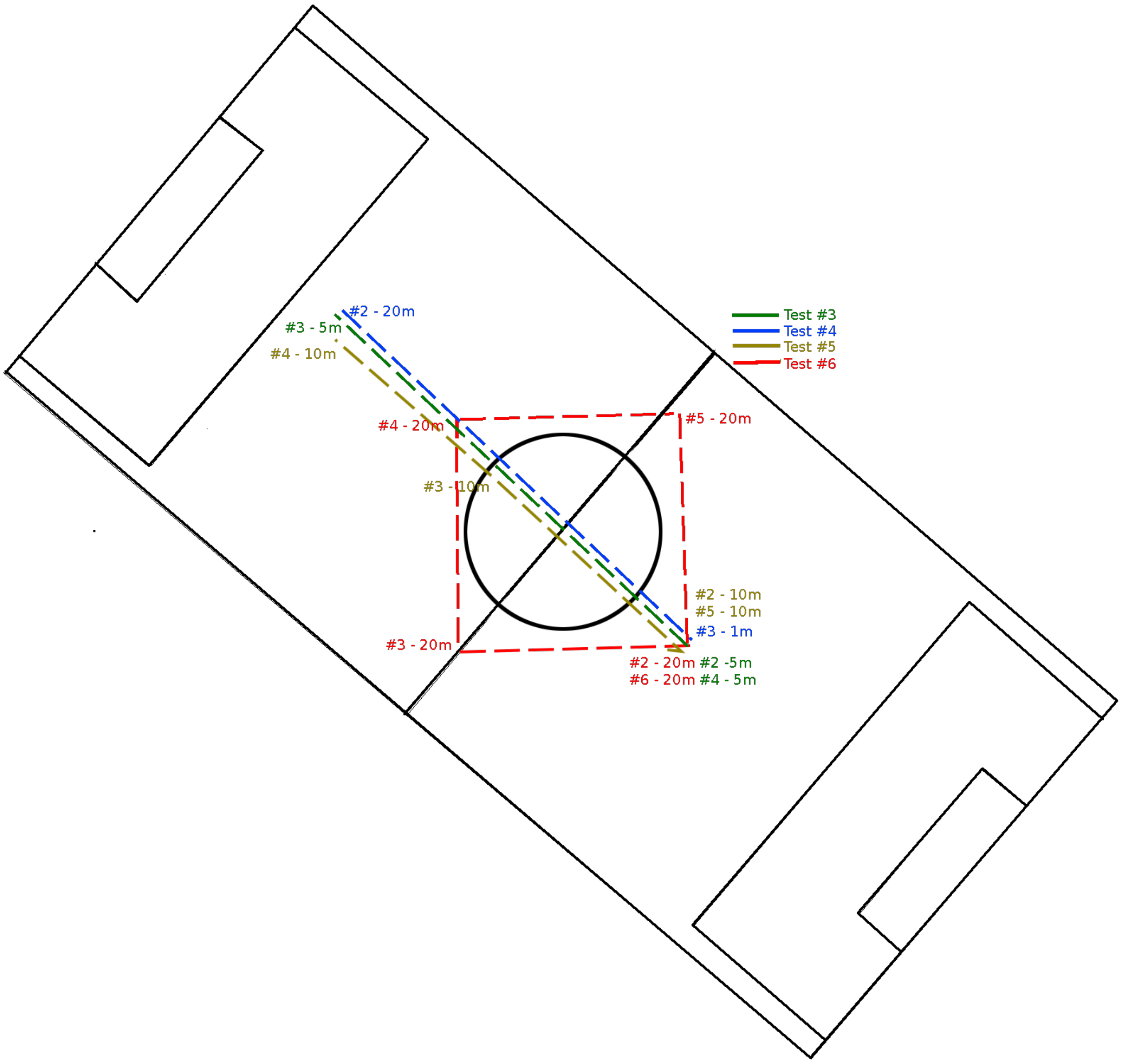


Figure 4.7: Drawing of steps in test #3-5. Does not include steps for landing and taking off. Green is test #3. Blue is test #4. Red is test #5.

Test #2

The aircraft moved to a height of 1-meter altitude and hovered, this was followed by increasing the altitude to 2 meters and from there directly landing.

Intermediate step	Average time (s)
All intermediate steps	5.40
Intermediate step after take-off	8.08
Intermediate step excluding take-off	4.37

Table 4.3: Averages of intermediate steps through all tests

Test #3

Relevant results of test 3 are that the aircraft used 10.24 seconds to the first point and 8.46 to move back. The same distance, and with both including a 180° turn.

Test #4

The aircraft flew 20 meters away while increasing altitude. It flew in a direct line to the designated point, at an angle to the ground. It also few at an angle to the ground to the home location.

Test #5

The aircraft used 4.94 seconds at step #3.5; this is with no change in direction or altitude.

Test #6

There is nothing to note in this test. The average time to turn 90° and fly 20 meters is 7.61 seconds, not including intermediate steps.

4.6 Discussion

There are several things to discuss, both in regard to the choices of implementation, the test, and the results of the tests. This section discusses each, in turn, starting with the choices of implementation, followed by the test as a whole, finishing with the results of the tests.

In regards to the implementation, the choice fell on using MissionControl from the iOS SDK. This choice was the one with the most freedom, considering that the FlightControl features are not supported. In this implementation, we did not use other actions than MoveTo, TakeOff, and Landing. As such, it might have been easier, development-wise, to use WaypointMission instead. This will be a consideration when implementing further features of the system.

The goal of the tests, as stated, was to see the capabilities of the aircraft and how it behaves in the air. As such, the tests are performed to find these capabilities. This means that although the tests performed were timed; they do not focus on the accuracy of these timings, nor the distances used in the test. The tests do not put too much weight on the accuracy; the goal of the tests was to get an overview of the capabilities rather than detailed specifications. The results of the tests would vary if performed under different conditions. And although time was measured under each test, the strength of these measures may be considered weak. They do, however give several indications to how the aircraft performs.

The results of each test show that the aircraft performs as expected—all tests completed from start to finish. The aircraft travels to designated points and does so in a direct line when altitudes differ. It also lands from any height given a land action. The most interesting result of the tests was the intermediate steps., averaging around 5.40 seconds between each action. This is a long time where the aircraft is not moving. The aircraft uses more time in the intermediate step directly following take-off than the other steps, but this still averages out to 4.37 seconds. This means that for each additional action performed; one could estimate adding another 4.37 seconds of flight time. For an entire flight covering an area, this is a lot. If the mission consists of 50 locations, then that is an additional 3 minutes 38.5 seconds where the aircraft is doing nothing. In the final implementation of automatic flight covering an area, each additional action comes at a high cost and should be avoided. This also includes actions that does not change direction or altitude, as shown by test #5.

As all the tests were done under heavy wind, there is reason to believe that the times would improve under better conditions. The aircraft was visibly fighting the wind to get to the right position, and this could be one of the reasons for the long intermediate steps.

4.7 Conclusion

The overall purpose of this chapter was to verify that the drone could fly automatically to predefined locations, how to implement this on a DJI Mavic 2 Enterprise, and testing the capabilities of the drone when flying automatically. DJI offers several SDKs, and the choice for this system fell on the iOS SDK due to the system requiring a user interface and the features of the Onboard SDK being available by communication through the Mobile SDK.

The implementation uses missions. This allows the developer to define actions for the drone to perform, such as taking off, flying to a location, and landing. Using these actions, one can define flight paths for the aircraft, and the aircraft will fly to them automatically.

The implementation was tested to chart the capabilities of the aircraft when flying automatically. The tests focus on basic flying, such as taking off, landing, and moving to locations, and changing altitude. The results of the tests show that a big factor for the time the aircraft uses is the intermediate steps between actions. This is time the aircraft calculates what to do next. Limiting the number of actions would reduce the overall completion time significantly.

Chapter 5

The application

5.1 Introduction

This chapter describes the implementation process of a mobile application displaying a map view. The goal is for the user to be able to define an area on a map. This area will represent the search area in the field where the drone should perform its search. The search area is then to be sent to a dedicated server, which calculates a suited path for the drone. The application will compose a mission based on the response and transfer this to the drone, which will then execute the mission.

The first sections of this chapter will give a quick introduction on how users can interact with a map and how a map can be implemented into an iOS application. Based on our problem, we compare map frameworks and find the one that is best suited. Next we dig into the implementation of the map in the application, before we verify our design through a number of usability tests.

5.2 Design considerations

A map can be used in many ways. It is important that the map's design respects and satisfy how a user is supposed to use it. A map can show roads and train tracks or it can show contours and hiking tracks. Which map representation to choose depends on the user's goal. If the map is interactive, possible controls should be visible for the user. This section describes design considerations when implementing a map view into an application.

5.2.1 Guidelines from Apple

Apple has implemented a number of *Human Interface Guidelines* [23] for Apple developers. These guidelines are specially designed for the development of iOS applications and ensure a more persistent user experience across different applications. The guidelines include a section [24] about how maps should be used in an application and how users should be able to interact with the map. The most important pinpoints are described below.

Keep the map interactive

Most maps found in mobile applications and websites are interactive. A map is interactive if the user can interact with the map, e.g., move the map, zoom in and out, and get directions. In a mobile application, one usually takes advantage of what is called gestures. Gestures are actions a user can perform on the screen using its fingers. Modern devices usually support both multi-finger touch and force-touch. This means that the number of fingers and the force of the touch respectively can be used to perform different actions. In other words, a soft press can trigger a different action than a hard press.

Consistency with the rest of the application

Both Google Maps and Apple Maps have their own layout and theme. The respective provider sets colors, buttons, and typography, and the map layout may not be consistent with the rest of the application. Apple states in their guidelines that developers should implement the app's theme into the map service. This ensures consistency and good user experience. One example is the color of annotations placed on the map by the user. If the annotation pin represents some data displayed to the user outside the map view, the pin's icon and color should be equally styled.

Keep map controls visible

It is common to add custom controls to a map. Custom controls can be used to interact with the map or another service in the application. To avoid custom controls to blend in with the map, one should choose colors wisely. Controls that look similar to objects on the map tend to be harder to see by the user. This can cause frustration and poor user experience.

5.2.2 Axismaps' Cartography Guide

Axismaps is a company formed in 2006 in USA. Their goal is to provide custom maps that conforms to the user's requirements. They focus on design and intuitive user interfaces, rather than algorithms. On their homepage we find a *cartography guide* described below.

Medium

When deciding on a map design, the medium in which the map will be displayed matters. On a higher level a decision has to me made whether the map will be displayed on paper or on a screen. A map on a paper is not interactive and can not be customized by the user in the field. All the information needs to be displayed on the paper, or it will loose its usability for the user. A screen can be either an ordinary display screen or a touch screen. Touch screens are usually smaller, but can often support a greater number of gestures, compared to a display screen. If the map should support several different media, maybe multiple designs have to be made.

Audience and Purpose

What the users are using the map for and *who* the users are play an important role when it comes to map design. Professional or enterprise users usually requires a more complex map design than ordinary users. For an ordinary user, a plain map with little interaction might be sufficient. Professional users, on the other hand, might have different requirements, based on their work and goals. The design can vary alot, based on profession and field of work.

Map-worthiness

Axismaps states that *just because data can be mapped, doesn't mean it should be mapped*. Maybe there are more suited ways of representing the data than on a map, e.g. in a table. You do not need a map if you know the area and have an address.

Interactivity

Static maps represents a cartography state at a given point in time, for example a printed map on a sheet of paper. Static maps can also be digital, often displayed as an image, but the level of interaction is still very limited, compared to what we call interactive maps. Interactive maps are often web-based and can be displayed on different digital media. They can hold more information than a static map and more control is given to the user. Careful consideration have to be given to the design, the flow of user experience, and the overall user interface.

5.2.3 A map to fulfill the user's needs

Most people with a smartphone have used a map application to find a place, give road directions, or locate oneself or others. A user always has objectives or goals he or she wants to achieve when using an application. If the application does not fulfill the user's needs, a more suited application will often replace it.

Just like an application is designed for some purpose, a map has to be designed to fulfill the user's needs. For example, if a person wants to use a map for road directions, it would not make sense to show railroads and hiking trails. A map with too much information, or wrong information, will be frustrating for anyone using the map.

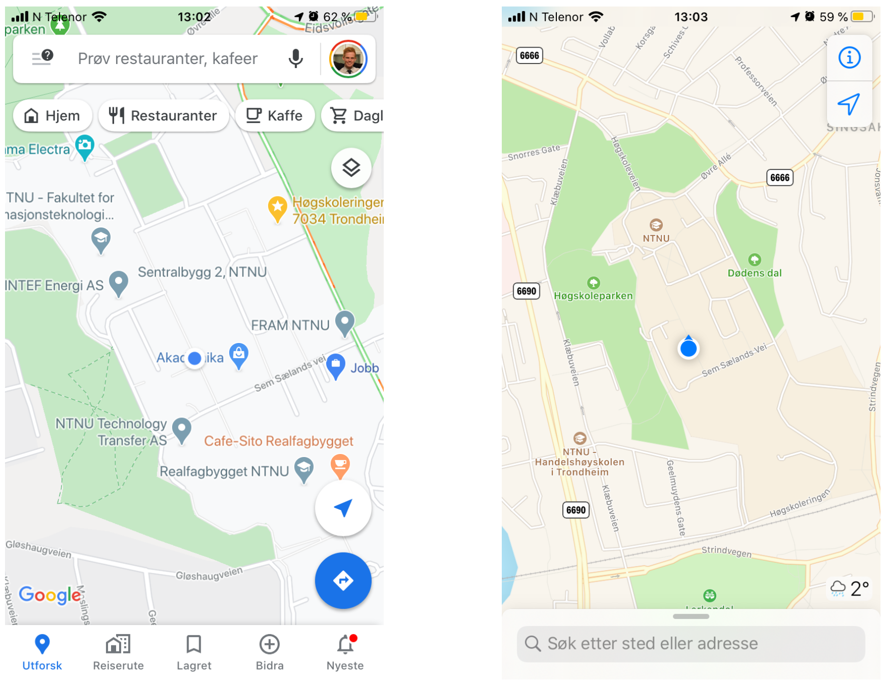


Figure 5.1: Comparison between Google Maps (left) and Apple Maps (right).

5.3 Framework

Digital maps have been on the market for a while now. Since this project concentrates on developing a mobile application for iOS devices, a requirement is that the map service can be implemented directly into an iOS application.

Two popular map services that fulfill this requirement are Google Maps [25] by Google and Apple Maps [26] by Apple. As seen in Figure 5.1, Apple Maps and Google maps look similar when it comes to the theme and the look of the map layer. On the other hand, we note that Google Maps offer a more rich layout, with more annotations, buttons, and controls. Other map service providers exists, like MapQuest [27], Open Street Map [28], and Maps.me [29], but to keep the implementation effort to a minimum, we focus on Google Maps and Apple Maps. Both of these services provide support for iOS and documentation on how to implement them.

5.3.1 Comparison

To take a decision on what map provider is best suited for the application, we need some evaluation criteria. First of all we require the service to be implementable into the iOS environment, either through a framework or an SDK (software development kit). Secondly, the map should support annotations (pins, markers), to be used as demarcations around the search area. It is a plus if the map can visualize the search area for the user in a good way, either as a coloured overlay, a

Criteria	**Google Maps**	**Apple Maps**
Can be implemented into iOS application	✓(Google Maps SDK for iOS)	✓(Apple MapKit Framework)
Map objects	✓	✓
Gestures	✓	✓
Ease of implementation	Moderate	Easy
Quality of documentation	Good	Good/moderate
Free?	✓(up to a certain number of requests)	✓
Storage/processing overhead	None/small	Moderate
Integration into iOS environment	Good/Moderate (separate SDK)	Good (Apple framework)

Table 5.1: Comparison between Google Maps and Apple Maps

polygon with corners at the annotations, or both. Other criteria are ease of implementation, rich set of gestures, intuitive action handling, the ability to customize the user interface, and add custom controls. The full comparison is summarized in Table 5.1.

Both of them have pretty much the same set of features and functionality. Some of Apple's documentation is outdated, but forums and development communities provide answers to common questions. Google Maps has a starter plan, which is free. Expenses will grow if you plan to use more advanced features and not be restricted by a request rate limit. Apple is free forever and have no rate limit. Since the application is implemented in an iOS environment, the implementation effort is smaller for Apple Maps than Google Maps.

Since Apple's MapKit is easier to implement and integrates better with the rest of the app, it is probably the most suited framework when it comes to implementation effort. On the other hand, after pilot testing the two frameworks we find that many of the required features are easier to implement with Google Maps. MapKit is also more restricted when it comes to gestures, especially drag-and-drop versus tapping. Based on this we found that Google Maps are more suited for the task and is more likely to fulfill the requirements of the mobile application.

5.4 Implementation

In this section, we will walk through the implementation of the map view step-by-step. To install Google Maps into our application sandbox we use *cocoapods*. Cocoapods is a dependency manager for Swift and Objective-C projects, much similar to *pip* for Python and *npm* for Node JS.

5.4.1 Privacy

To use the GPS location of the device the user has to allow this in the application. This is handled by iOS under the hood, but we need to provide a text that describes what the location is used for. We can choose to ask for location *All the time* or *When in use*. For simplicity, we use both of them. These global settings are configured in a special file, called *info.plist* (information property list).

5.4.2 The Map View

By using XCode Storyboards we simply drag and drop a MapKit View element into the storyboard (see Figure 5.2).

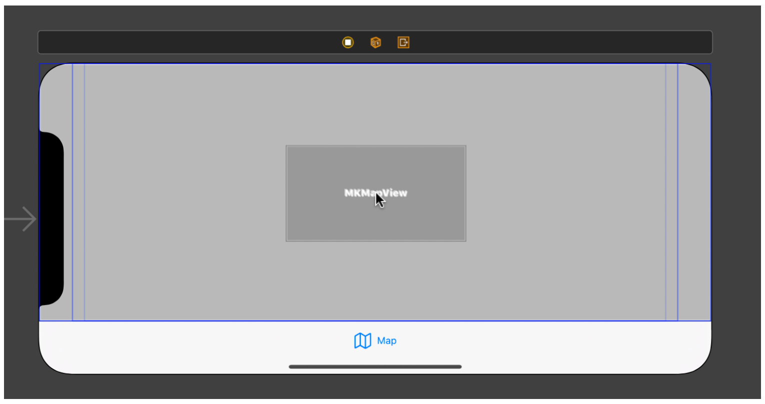


Figure 5.2: Map view: Dragging the MapKit View into the storyboard

Since a mobile application can be used on variety of devices with different screen sizes, we find it best to let the map view take up all the space on the screen. We do this by defining *constraints* on the views. In our case we constraint the view to have a distance of zero to the screen in all directions.

5.4.3 The Map Controller

A controller is responsible for controlling what is being displayed in the view. To make changes to the map, we define a Swift class called *MapController* and set the map as the view for the controller. With a reference to the map view, we can handle gestures, add objects to the map, or change the layout. Listing 5.1 defines a basic map controller.

```
class MapController: UIViewController {
   // Reference to the map view
   @IBOutlet weak var mapView: MKMapView!

   // List of annotations placed on the map
   private var annotations: [CustomPointAnnotation] = []

```

```
    @objc private func handleMapTap(gestureRecognizer:
        UITapGestureRecognizer) {
        // Get the tapped location
        let locationInView = gestureRecognizer.location(in: self.mapView)
        // Convert it into coordinates
        let coordinates = self.mapView.convert(locationInView,
            toCoordinateFrom: mapView)
        // Create a new annotation and append it to the annotations array
        let newAnnotation: CustomPointAnnotation =
            CustomPointAnnotation(coordinate: coordinate, index:
            self.annotations.count - 1)
        self.annotations.append(newAnnotation)
        // Display the new annotation on the map
        self.mapView.addAnnotation(newAnnotation)
    }
}
```

Code listing 5.1: Swift implementation of a MapKit MapController.

New markers are added to the map when the user taps the view. When this happens, we also update the polygon with a new corner. The polygon is defined by a path between the marker positions. Every element is connected consecutively to each other, and the last element is connected to the first to form a closed graph. In Figure 5.3 we see the polygon, created by six taps on the screen.

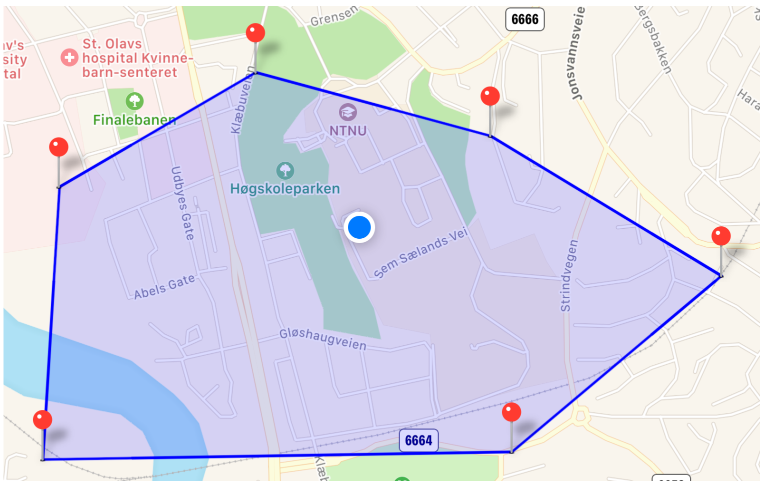


Figure 5.3: Making a polygon shape on the map. Red pins represents the outer bounds of the polygon. Blue marker represents the user's location.

5.5 Test

We implement a test to verify that the user interface works as intended. To make a decision between Google Maps and Apple Maps, we execute the same test for each of them. The two main goals for this test are as follows:

- Validate the map design and functionality
- Discover which map provider that gives the best user experience

Even though the map is just a small part of the application, we perform an isolated usability test to validate and detect possible faults with the design. By isolation the functionality in this manned, the user is not biased by other features. This gives a better foundation for taking further actions on the design of this particular feature. Usability tests are performed by non-biased users, who have never seen the application before. This simulates how the application would work in real life – when a user installs it and use it for the first time.

5.5.1 Preparations

It is important to be well known with both the product itself and the test. [30] We execute pilot tests in advance to discover errors in the application and the test. This ensures that the application does not crash during the test and the user is able to complete the tasks. Then we define test objectives based on various user journeys. These objectives make up the tasks the test subject will try to perform during the actual test. For each objective we define parameters – both qualitative and quantitative – from which we extract the test results. We define a numeric score for each parameter whenever this is possible, which will give a good measure for comparison.

5.5.2 Objectives

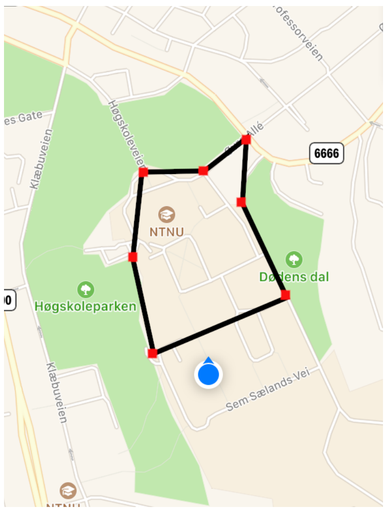
Table 5.2 lists the test objectives. Each objective has one or more parameters as a measure on the success of the objective.

Test Objective	**Objective parameters**
O1: The user should be able to define a polygon on the map, based on a predefined area given to the user	• Time to complete task • Polygon shape accuracy (similarity to predefined area)
O2: The user should be able to change the polygon, according to a new predefined area given to the user	• Time to complete task • Number of trials • Number of drag and drops on the polygon corners • Polygon shape accuracy (similarity to predefined area)

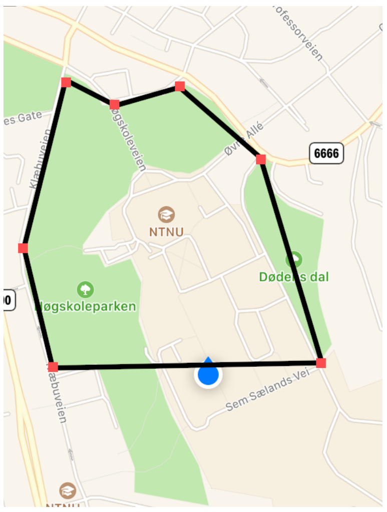
Table 5.2: Test objectives

5.5.3 Test implementation

To simulate a farmer who is familiar with the area and know exactly where the search area should be defined, we draw a map in advance (see Figure 5.4a). The test subject's objective is then to replicate this drawing as good as possible. For O2, the objective is to change the shape accordingly to fit the shape in Figure 5.4b.



(a) Shape to be drawn by test subject in O1.



(b) Shape to be drawn by test subject in O2 after editing.

Figure 5.4: Maps for test.

The design does not give any description on how to perform an action, so the user will have to figure this out by himself. The key thing to investigate is if the test subjects are able to use prior knowledge about maps in mobile applications to figure out how to complete the task. The test will show whether it is Google Maps or Apple Maps that provided the most user friendly and intuitive design.

5.5.4 Test results

Chapter 6

Altitude

There are few places on earth that are completely flat. In Norway, especially, there are many changes in altitude. About two-thirds of the land area of Norway is mountainous. A lot of the animals grazing in Norway graze in uneven and drastically changing terrain. This project aims at creating a fully automatic coverage of an area using a drone. This means that along a path, the altitude of points along the path will most likely differ significantly. Therefore two problems need to be addressed in regards to the terrain of an area.

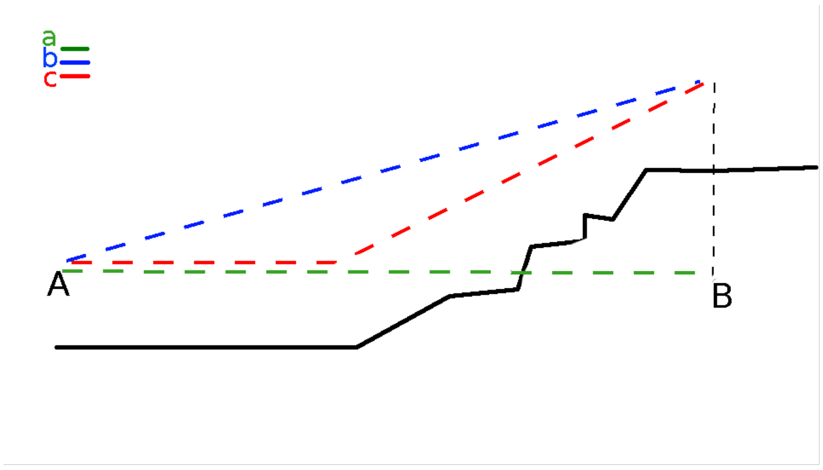
1. Obstacles due to differences in altitude. See figure 7.9b for an illustration highlighting this problem.
2. Field of view of the camera due to difference in altitude. See figure 7.9c for an illustration highlighting this problem.

With regards to obstacles, this mostly involves an increase in the terrain leading to a hill or mountain getting in the way of the drones' flight path. When it comes to the field of view of the camera, pictures should be taken at an optimal altitude to ensure the balance between a large area covered and the detail of the image. These two problems mean that the final path needs to consider the altitude of each point in the path, *as well as points between*.

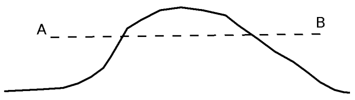
As a path is defined as a series of points, the terrain may differ significantly along a line between two points. The altitude will have to be added to each of these points. However, this does not consider the terrain between each point in the path. The altitude of intermediate points at a defined interval along the line needs to be added as a consideration. Figure 7.9a shows such a terrain difference. While the blue and red line goes to the same point, the red line follows the terrain better due to having an intermediate point considering the altitude.

6.1 Height data in Norway

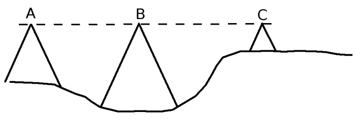
As altitude needs to be considered when calculating the final path, the system should have access to height data of the area, and use the data to add altitude parameters to the path. This section shows the different height maps available



(a) Shows different paths from a point 1 to point B. Green line disregards altitude completely. Blue line goes directly from point A to point B with the proper altitude increase over point B. The red line uses a intermediate point to keep a path that deviates minimal from the altitude relative to the terrain



(b) Shows the issue of disregarding terrain in regards to hitting objects. Also shows the need for intermediate points.



(c) Shows the issue in differing altitudes with regards to the area covered by the camera. Point A shows the ideal height. In point B details are lost in the picture due to the height, while point C inefficiently covers a relatively small area.

Figure 6.1: Illustrations regarding difference in altitude of the terrain

in Norway and how they are added to the system. There are several height maps available in Norway. These depict the terrain in meters above sea level. They differ in accuracy and method of collecting, as well as some being a collection of multiple other data. All data is publicly available at *hoydedata.no*. [31] The data can be categorized into two: *Digital Terrain Model (DTM* and *Digtital Surface Model (DOM)*.

6.1.1 DTM

DTM is height data collected depicting the natural surface of Norway, it does not consider obstacles such as trees or human-made structures. NN2000 is the current standard for altitude measurements in Norway. [32] It started in 2011 and was finished in 2018. It replaced NN1954, due to changes in the height of Norway as a whole. Norway rises slightly. NN2000, although started in 2011, uses the heights of the country in 2000. Later, land rising is modeled and applied to these data; this means NN2000 should be accurate for the foreseeable future. DTM currently covers the entirety of mainland Norway.

6.1.2 DOM

DOM is height models considering obstacles in the terrain. It includes trees and man-made structures and is usually made from Local Point Clouds. Local point clouds are part of the *Nation Detailed Height Model (NDH)*. [33] Point clouds are models of points in a 3D-space. When used as height models point clouds are detailed height measurements of the terrain in an area. NDH, as a project, aims to create a highly detailed model of Norway covering 253 000 km^2. It uses measurements from planes or helicopters with mounted laser scanners. The goal is an accuracy of 1mx1m. Local point clouds are available, but due to the size of the data it has to be downloaded as separate files for each local region. It is possible oto get the height measurement of a single point, given that there exists a measurement of that point. DOM is not complete for the entirety of Norway.

6.1.3 Conclusion

While the system could use both data sets, DTM was chosen due to currently being complete for the entirety of Norway. DOM is not complete for Norway yet. The system in this project should be usable in most of Norway, especially in areas where animals graze, and DOM coverage in these areas is still lacking. The goal of this system is to cover and area and find objects of interest. As the focus lies on grazing animals that most of the time is at the terrain and not on top of human-made structures or trees, there does not need to be a camera consideration that justifies using DOM. However, it may be used to check that the path of the drone does not cross any obstacles. Therefore the final system also uses DTM in areas where it is available to make sure there are not any unforeseeable objects in the path.

6.2 Coordinates - SRIVES OM! SkRIV OM UTM OGSÅ, KANSKJE IMPLEMENTASJON

The algorithm used to define a path for covering an area is given coordinates in the form of longitudes and latitudes, and return a path with the same type of coordinates. However, as the algorithm gives points for the drone to follow and the earth being a globe, some challenges arise.

6.2.1 Great circle

Great circle in regards to navigation is the practice of using the shortest line between two given points.[34] The line follows the globe around its curvature and ends up at the same point. Any two points on the earth's surface have a unique great circle except points being directly on opposite sides of the globe; they will have an infinite amount of great circles. Using the great circle, one can easily find intermediate points between two other points. Using Great Circle navigation, the system ensures the drone takes the shortest path possible between points, while still following the curvature of the earth. Figure 7.10 illustrates a great circle derived from two points *P* and *Q*.

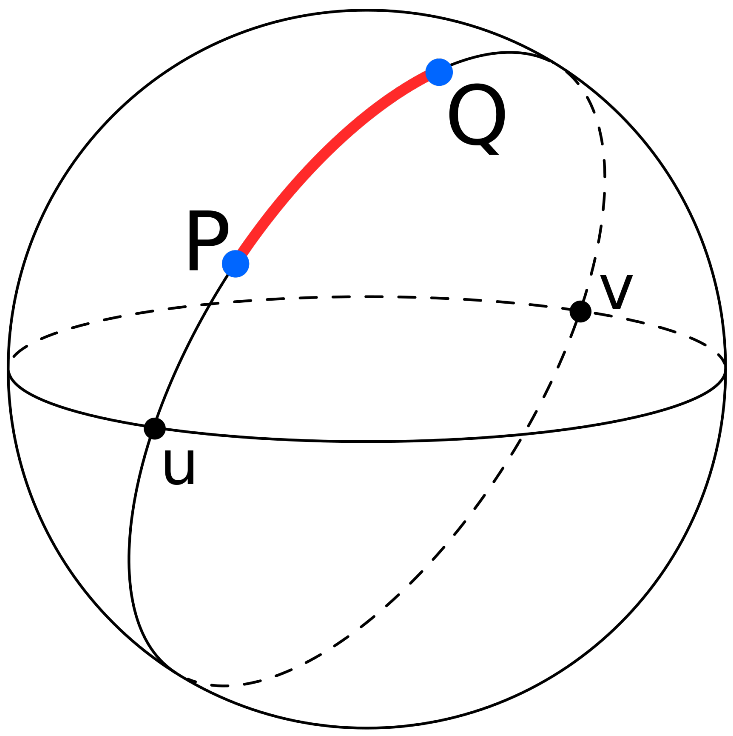


Figure 6.2: Illustration of a great circle line made from points P and Q. "*Illustration of great-circle distance*" *by CheCheDaWaff, licence: CC BY-SA [35]. No changes made.* `https://upload.wikimedia.org/wikipedia/commons/c/cb/Illustration_of_great-circle_distance.svg`

6.3 Camera

The altitude relative to the terrain of which the aircraft should fly depends on the purpose of the area coverage. If the purpose is spraying pesticides on a field, the height of the aircraft will be lower than if the purpose is to create a birds-eye view map of the area. As this project aims to create a solution for finding grazing animals, the height at which the aircraft should fly above the terrain depends on the camera mounted on the drone. Aspects of the camera need to be taken into consideration, such as field of view (FOV), sensor, aperture, and focal length. The goal of this section is to find the properties of a camera and use this top to find how large a picture covers an area from a given height.

6.3.1 Aspects of Cameras

As previously stated, several aspects of cameras need to be taken into consideration when determining the altitude of the aircraft. These aspects determine how well things are in focus, the detail of the image, how much of a scene is in the image, and more. Each relevant aspect is explained in this section, as well as how it affects the altitude the aircraft should travel.

Aperture

Aperture is the opening on a camera lens where light passes through.[36] Aperture affects two things in an image, *brightness (or exposure)* and *depth of field*. The brightness of the image increases with a large aperture and decreases with a small one. It is comparable with how the human eye works. In a bright room, the pupils constrict to minimize the light hitting the retina, while the opposite is true for a dark room, where the pupils dilate to let in more light. One would want a large aperture if the environment is dark and a small aperture if the environment is bright.

The other thing aperture affects, is the depth of field. With a large aperture, both the foreground and the background gets blurred relative to the focus point. With a small aperture, the image is mostly clear both far away and close.

Aperture is depicted as f/x, where x is the inverse size of the aperture. As such, a small x, such as f/1.4, means the aperture is large, and the image is brighter and more unfocused in the foreground and background. While an aperture of f/16 means a small aperture that is less bright but has a clear foreground and background.

Aperture needs to be considered as the depth of view determines how detailed objects of varying distances will be in the image, as well as the lighting of the environment.

Focus

Focus is directly correlated to the depth of the field. At the *plane of focus* an object is sharp and detailed.[37] The plane of focus is the distance where the objects are at optimal sharpness. Objects closer to and further away from the plane of focus get exponentially more blurry. Most lenses can adjust the focus so that the desired object is in the range where the image is still sharp. Modern cameras have auto-focus capabilities.

The drone has auto-focus, and as such, this does not need to be considered.

Sensor

The sensor of a digital camera is what actually captures the image.[38] It is a chip consisting of millions of light-sensitive elements called pixels. The light-sensitive elements translate the incoming light into digital values. Sensors differ in several aspects, but the two most important to this project, are sensor size and amount of pixels.

The sensor size determines the actual size of the sensor. The standard size is 36x24mm. With the same lens, a smaller sensor will show a subset of the image of a larger sensor; this is because the lens will focus the light around the smaller sensor, where a larger sensor will cover more of the light. As such, the lens used on a camera will have different properties depending on the sensor size. A smaller sensor will have a smaller field of view with the same lens as a larger sensor.

The amount of pixels determines the resolution of the resulting image. As each pixel is converted to a digital value, more pixels means images with a higher level of detail. The number of pixels is usually denoted as megapixels, meaning the current cameras has a magnitude of millions of pixels.

The sensor is relevant to this project because it determines how much information the image holds, as well as the field of view.

Field and Angle of view

The *field of view (FOV)* is the section of the world observed from the camera.[39] *Angle of view(AOV)* is the angle determining the size of the field of view. A small angle of view, creates a narrow "cone" of what will be seen, while a large angle of view means more of the scene is covered in the image. Figure 7.11 shows how the angle and distance determine the size of what is covered in the scene and the difference in size depending on the distance. The angle of view depends on the sensor size and the focal length of the lens on the camera.

In photography, angle of view and field of view are often interchangeable, where the field of view is the most commonly used and is given as an angle. In this paper, for the sake of clarity, AOV is the angle, and FOV is the resulting coverage of the scene.

As images are mostly rectangular, the angle of view is different horizontally and vertically. As such, it is common to describe the different values of the angle of view as *horizontal angle of view* (HAOV) and *vertical angle of view* (VAOV), and *diagonal angle of view*. The same for the different values of the field of view, respectively: *horizontal field of view* (HFOV) and *vertical field of view* (VFOV), and *diagonal angle of view* (DFOV).

To calculate the angle of view the focal length of the lens and the sensor size is needed. The formula 7.4 calculates the angle of view where *s* is the sensor size,

f is the focal length, and θ_{AOV} is the angle of view.

$$2arctan(\frac{s}{2f}) = \theta_{AOV} \tag{6.1}$$

Given the angle of view and distance, the coverage of the resulting image can be calculated. The formula for finding the field of view given the angle of view is shown in 7.5, where θ_{AOV} is the angle of view of the camera, l is the distance to the scene, and FOV is the resulting field of view.

$$2tan(\frac{\theta_{AOV}}{2}) * l = FOV \tag{6.2}$$

It is also relevant to calculate the angle of view of one direction, given the angle of view of the other direction and the ratio of the sensor. Formula X and X show how to convert between HAOV and VAOV, where θ_{HAOV} is the horizontal angle of view, θ_{VAOV} is the vertical angle of view, and R is the ratio ($\frac{H}{V}$) of the sensor.

$$2arctan(\frac{tan(\frac{\theta_{HAOV}}{2})}{R}) = \theta_{VAOV} \tag{6.3}$$

$$2arctan(tan(\frac{\theta_{VAOV}}{2}) * R) = \theta_{HAOV} \tag{6.4}$$

To find the diagonal angle of view, we first need to find the length of the diagonal on the sensor. The relationship between the diagonal and the length of each side is shown in formula 7.8

$$h^2 + v^2 = d^2 \tag{6.5}$$

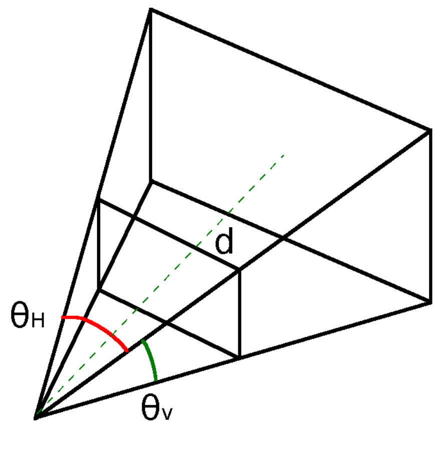


Figure 6.3: Figure showing how the angle of view and distance determine the area covered by an image

6.3.2 Specifications

The camera specifications of the DJI Mavic 2 is shown in table 7.1 and are taken directly from the official manual REF. The specifications are both for the thermal and the visual camera. Some things to consider when looking at the specifications for the thermal camera are:

- The sensor size and lens are not given. However, the sensor resolution is given.
- The the HFOV is given as an angle; this is the angle of view as defined in section 7.6.1. The horizontal direction is also the only one given.

We can use the sensor resolution and the horizontal angle of view to find out the ratio and from this calculate the vertical angle of view of the thermal camera.

We get more information on the visual camera. Some things to consider when looking at the specifications for the visual camera are:

- 1/2.3" CMOS has a ratio of 4:3 REF.
- The effective pixels of 12M means there is 4056×3040 pixels REF.
- The direction of the FOV (angle of view) is not given.
- "35 mm format equivalent: 24 mm" means that the lens has the same focal length as a 24 mm lens on a 36x24 mm sensor.

Since the angle of view does not have a direction, it will need to be calculated using the focal length and sensor size.

The lens is not able to zoom. The specifications are sufficient to calculate the area covered, given a specified height.

Thermal camera	
Sensor	Uncooled VOx Microbolometer
Lens	HFOV: 57° Aperture: f/1.1
Sensor resolution	160x120
Visual camera	
Sensor	1/2.3" CMOS Effective pixels: 12M
Lens	FOV: approx. 85° 35 mm format equivalent:24 mm Aperture: f/2.8 Focus: 0.5 m to ∞

Table 6.1: DJI Mavic 2 Enterprise Dual camera specifications

6.3.3 Calculations

For this project, the coverage of an image from a given height is needed. The angle of view in both the horizontal and vertical directions can be used, with the height, to calculate the coverage. Using the specifications given in 7.1 we can use the formulas from 7.4-7.8 to find all relevant values. We need the AOV both horizontally and vertically for both the thermal and visual cameras.

Starting with the thermal camera, we already know the horizontal angle of view from table 7.1 as 57°. The sensor 160x120 meaning the ratio is 4/3. Using formula 7.6, we get a vertical angle of view of $\tilde{4}4.31°$.

Finding the horizontal and vertical angle of view of the visual camera is more difficult, and some assumptions have to be made. The FOV from the manuals does not give a direction. Using formula 7.4 and a sensor of 36mmx24mm, results in a diagonal of 84.11°. This is close to the approximation of 85°given in the manual, and it can be assumed that this is the diagonal angle of view.

The problem with this value is that it is a result of having a sensor with an aspect ratio of 3:2. The actual sensor has a ratio of 4:3. The most likely reason for this discrepancy is that all the information on the lens follows the 35 mm equivalent, and this has a sensor ratio of 3:2.

To convert the angle of view given a 3:2 ratio to the horizontal and vertical angle of view given a 4:3 format we can change the sensor size of the 35mm equivalent to 36mmx27mm, which has a ratio of 4:3, and use this with formula 7.4. Using a f of 24mm and s as the sensor size for a given direction we get:

- Finding the vertical field of view:
  $2arctan(\frac{27mm}{2*24mm}) = 51.72°$

- Finding the horizontal field of view:
  $2arctan(\frac{36mm}{2*24mm}) = 73.74°$

As we now have the angle of views of both the thermal and visual camera, we can calculate the field of view in both directions given a height.

6.3.4 Implementation

As the different angle of views of both the thermal and visual camera is known, they can be used to find the field of view. Code listing 7.1 shows the implementation for finding the field of view. Formula 7.5 was translated to Javascript code and can be found on line 6 in code listing 7.1. Using function *getFov* an object with all field of views is returned for a provided height. Table 7.2 shows the different field of views given any height.

```
1  const degreesToRadians = (degrees) = {
2     return degrees * (Math.PI/180);
3  };
4
5  const fov = (angleOfView, height) => {
6     return 2 * Math.tan(degreesToRadians(angleOfView) / 2) * height;
7  };
8
9  const aov = {
  // Thermal
  thermal: {
    horizontal: '57',
    vertical: '44.31'
  },
  // Visual
  visual: {
    horizontal: '74.74',
    vertical: '51.72'
  }
};

const getFov = (height) => {
  return {
    thermal: {
      horizontal: fov(aov.thermal.horizontal, height),
      vertical: fov(aov.thermal.vertical, height)
    },
    visual: {
      horizontal: fov(aov.visual.horizontal, height),
      vertical: fov(aov.visual.vertical, height)
    }
  };
}
```

Code listing 6.1: Javascript implementation for finding the field of view of the drone given a height

Height / **Camera**	10 m	50 m	100 m
Thermal	10.86x8.14 m	54.30x40.72 m	108.6x81.4 m
Visual	15.27x9.69 m	76.37x48.47 m	152.7x96.9 m

Table 6.2: Shows field of views of the thermal and visual camera at different heights

6.3.5 Conclusion

By looking at the components of a camera, like a sensor and a lens, the properties of the camera can be found. Formulas for finding the angle and field of view were used with the camera specifications from the DJI Mavic 2 Enterprise. After finding the angle of view of both the thermal and visual camera, a simple Javascript program was implemented that can be used to find the field of view of the camera on the drone given a particular height.

Chapter 7

Flight path

In this chapter we aim to investigate multiple models for a flight path covering a specified search area. The goal is to produce a flight path that covers the area efficiently and with high accuracy. The problem can be divided into two sub-problems. First, a flight path has to be defined in the x/y-plane. This path makes up the point-to-point coordinates for which the drone should fly during its search. Second, we need to take into consideration the drones height relative to the ground. The drone does not record the altitude above sea level, nor the altitude above the ground (except for low altitudes). The objective is to find the height above the ground. This will make it possible to keep the drone at a constant altitude above the ground by adjusting the drone's relative altitude to the starting point.

7.1 Introduction

We find drones as a suited tool in missions within inhospitable environment. They can perform automatic, semi-automatic or remote controlled flights in places in which can be dangerous for humans to operate. For the drone to be able to autonomously find its way in such places, *path planning* has to be implemented into the aircraft's flight controller. Path planning is the process of finding a valid sequence of actions to get from point A to point B in a given environment. The environment can be observable, partially observable or not observable.

The drone usually has a sensor or camera. The area on the ground captured by any such device is called a footprint. To cover the entire AOI, the size of the footprint has to be considered when producing the flight path. Figure 7.1 is an example of a distribution of footprints to capture information about the whole search area. If the aircraft is equipped with a camera, the footprints are images of the ground, taken from a bird's-eye point of view.

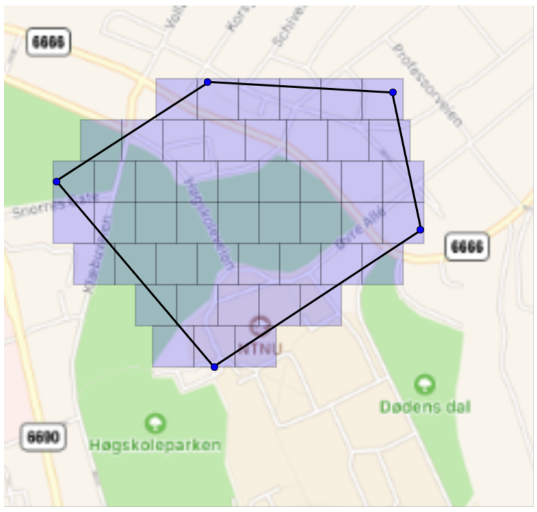


Figure 7.1: Footprints covering search area. Black lines represents the search area. A light blue square represents a footprint.

7.2 Coverage path planning

Path planning can be sufficient in many cases. But when the targets are not defined as discrete locations, path planning algorithms come to short. This applies for instance if we are interested in investigating an area of ground instead of a single point. The coverage path planning (CPP) problem is defined as finding a path that completely covers an area of interest (AOI), based on restrictions and a description of the environment. In the following we will investigate how the area of interest is defined and what its restrictions are. Next we define metrics for a comparison before we review different techniques for how the problem can be decomposed into smaller sub-problems.

7.2.1 Area of interest

The AOI is the area where the drone performs its search. As defined in Equation 7.1 the AOI is represented by a sequence S of vertices, where each vertex v_i is defined as a pair of coordinates (v_x^i, v_y^i).

$$S = \{v_i = (v_i^x, v_i^y)\}_{i=0}^{N-1}, \tag{7.1}$$

where $N = |S|$ is the number of vertices. An edge e_i is the line between vertices e_i and e_{i+1}, where $0 \leq i \leq N-2$. Given vertex v_i and its neighbouring vertices v_{i-1} and v_{i+1}, we have that v_i's internal angle θ_i is given by Equation 7.2.

$$\theta_i = \arccos \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}, \tag{7.2}$$

where $\vec{a}$ and $\vec{b}$ are vectors $[v_i^x - v_{i-1}^x, v_i^y - v_{i-1}^y]$ and $[v_i^x - v_{i+1}^x, v_i^y - v_{i+1}^y]$ respectively.

Furthermore, we restrict our AOI to be simple, closed polygons. To simplify our notation, given a vertex v_i, we denote the next vertex in the sequence as $v_{next(i)}$. Since the sequence now represents a closed polygon, we have that $next(i) = i(modN - 1) + 1$. It follows that $e_i = v_{next(i)} - v_i$. Figure 7.2 shows an example of an AOI with vertices, edges and internal angles labeled accordingly.

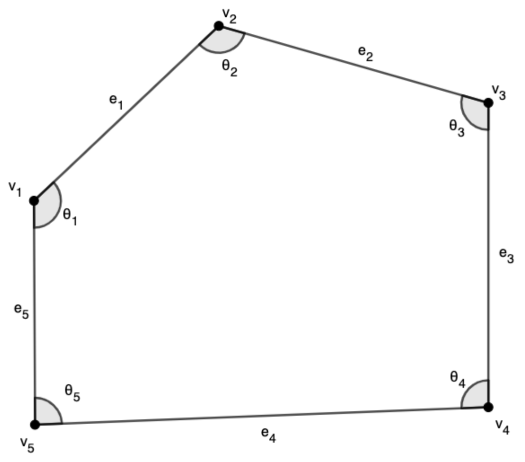


Figure 7.2: Example of AOI polygon with labels.

It is important to define how the AOI will look like. Different methods and concerns will apply, depending on the shape of the AOI. [40] has reviewed different flight paths for rectangular AOI. [41] investigate different sweeping angles relative to the wind and extends the shape to convex polygons. More complex methods will apply when also considering concave methods. In the following we will address different properties the AOI can have, by concentrating on polygonal properties.

Polygons

From the design of the application, we focus on polygon-shaped search areas. This makes it easier for the user to define the desired area on the map. In addition, it is feasible to limit the shape to polygons when defining the flight path. From 4 we know that directional changes in the flight path is an expensive operation for the aircraft. Because of this we aim to minimize the number of turns the aircraft makes during the search mission.

Polygons can be classified based on various properties. In the following we define important properties and combinations of them.

A polygon is either convex or concave. In practice, if a polygon is convex, any straight line drawn through the polygon will only intersect two edges. It also means that one can move in a straight line between any two points inside the

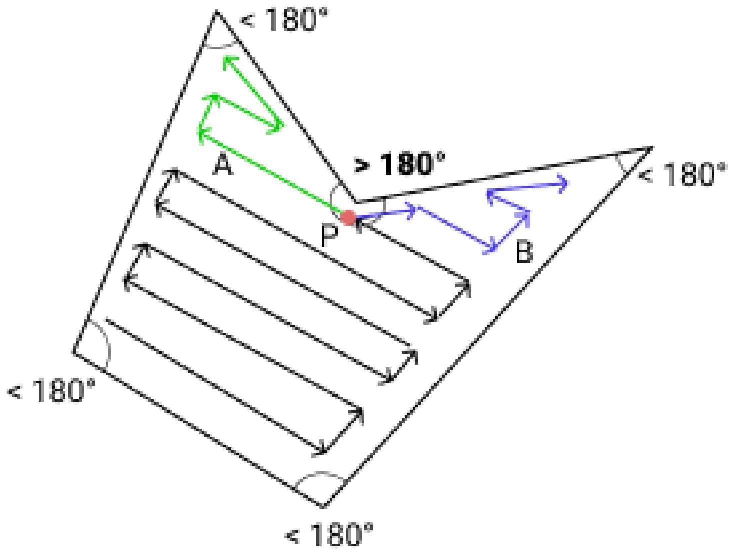


Figure 7.3: A concave polygon with a path planning problem.

convex polygon, without crossing any edges. Theoretic definitions are given below.

Definition 1: Convex Polygon
A convex polygon is a polygon with all its interior angels being less than 180°. See Figure 7.4a.

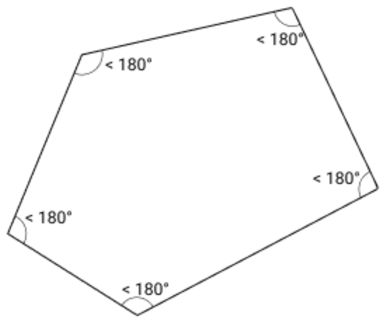
Definition 2: Concave Polygon
A concave polygon is a polygon with one or more interior angels being greater than 180°. See Figure 7.4b.

As seen in Figure 7.3, a concave search area brings more complexity to the path planning problem. At point P, the planner have to make a decision whether to choose path A or path B. Regardless of which is chosen, the aircraft will have to follow a path back to P to cover the other area. Hence, the drone will fly over an already covered area, which is undesirable.

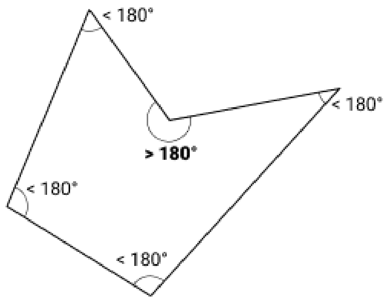
We classify polygons as simple or complex, based on the vertices' relative position. If vertices in a polygon are aligned so that edges intersect, we say that it is a complex polygon. Otherwise it is simple.

Definition 3:
A polygon is complex iff two or more of its edges intersect. Otherwise the polygon is simple. See Figure 7.5.

Complex polygons adds more complexity to the path planning problem. To simplify the path planning task operation we require the search area polygon to be non-complex. Hence the application will have to make sure the user only submits simple polygons.

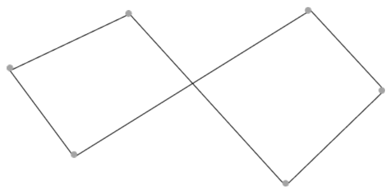


(a) Convex polygon



(b) Concave polygon

Figure 7.4


(a) Complex polygon

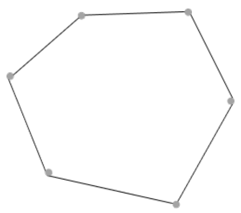

(b) Simple polygon

Figure 7.5

A polygon is *equiangular* iff all its angles are equal in measure. A polygon is *equilateral* iff all its sides are having the same length. A polygons regularity depends on these to properties and defined below.

Definition 4:
A polygon is regular iff it is both equiangular and equilateral, hence all angles and sides are equal respectively. If one or more of these conditions fail to hold, the polygon is irregular.

7.2.2 Performance metrics

Proper performance metrics must be defined when investigating different solutions to the CPP problem. What performance metrics to consider may depend on the environment where the aircraft performs its search, such as the shape and size of the AOI, the presence of no-fly zones within the search area and the type of aircraft. For instance, area coverage might be more important in some applications than in others.

The first thing we need to consider is the area coverage, that is how many square feet of ground the aircraft can cover. This value depends on other properties, such as the drone's altitude, the pitch of the camera gimball, and the cameras focal length. If these properties are constant, we can compare different approaches, by measuring the area that the drone covers. For better comparisons, we use the area coverage, expressed as a percentage of the total AOI. A high area coverage is desirable. On the other hand we do not want to cover an area more than once.

Another metric is the amount of time it takes for the drone to complete its mission. It follows from the results in Section 4.5.2 that the time used by intermediate steps is significant. Hence, the flight time will increase as the number of intermediate steps increases.

This brings us to the third metric, which is the number of operations. The literature [42, 43] agrees that it is desirable to minimize the number of turns. This is because the flight time increases as the drone has to make more stops and accelerations during its flight.

Another useful metric is the average ground speed. This is calculated by dividing the total distance traveled by the total time spent executing the mission.

For the rest of this section we will focus on finding a solution to the CPP problem that completely covers the AOI and minimized the number of turns. At the same time we aim to completely avoid overlapping paths, i.e. not cover areas more than once.

7.3 CPP in Convex Polygons

As done by [43], in the following we define the width of a convex polygon:

Definition 5:
Width of convex polygon The width (W) of a convex polygon (P) is the minimum span (D) between two parallel lines of support (l_1, l_2).

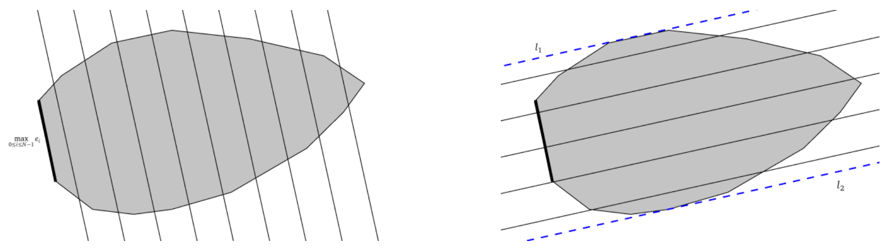
l_1 and l_2 are parallel lines that intersects the polygon at opposite sides of the convex polygon P, so that P lies to the left (or right) of l_1 and to the right (or left) of l_2. It is given by intuition, and mathematically proven by [43], that at least one of the support lines of a minimum span lies 'on top of' a edge of P.

7.3.1 Sweeping direction

Di Franco and Buttazzo proposes a solution using back-and-forth (BF) motions. The authors claims that a high energy efficiency is achieved by setting the scanning direction to be parallel to the longest edge in the polygon. Jiao *et al.* on the other hand, flies along the vertical direction of the width, that is, parallel to the lines of support.

Figure 7.6 demonstrates these two flight directions on a convex polygon. To simplify, the paths from one sweep to another are not included. Furthermore, if we count S sweeps across a polygon we will have a minimum of $S \cdot 2 - 2$ turns to cover the entire area.

A flight direction parallel to the longest edge (7.6a) results in more sweeps across the polygon than using the direction of the support lines (7.6b). We count 9 sweeps and 16 turns for 7.6a and 5 sweeps and 8 turns for 7.6b. Hence, when



(a) Flight direction parallel to the longest edge.

(b) Flight direction parallel to the lines of support.

Figure 7.6

it comes to the number of turns, a flight direction parallel to the support lines is preferred over a direction parallel to the longest edge.

A flight direction parallel to the support lines is generally the best solution to minimize the number of turns. To prove this we assume that the number of turns is proportional to the number of sweeps. In fact it is also proportional to the width of the polygon. Next we define the number of sweeps in Equation 7.3.

$$S = \lceil W/L_x \rceil, \tag{7.3}$$

where S is the number of sweeps, W is the width of the polygon and L_x is the width of the footprint. $\lceil x \rceil$ denotes the ceiling function, which maps a number x to the least integer greater than or equal to x. Since L_x is a constant and W is at its minimum, we have that the number of sweeps S has reached its minimum.

7.4 Convex Decomposition

Convex decomposition is a popular method to reduce the CPP problem into smaller individual parts. It takes a (possibly concave) polygon as input, performs a decomposition (if necessary) on the polygon and outputs one or more *convex* polygons. The motivation for a convex decomposition is the assumption that it is easier to find a covering path for each of these convex polygons than the complete AOI.

Simple, non-complex and convex AOIs do not require any decomposition.

Algorithms for convex decomposition are proposed in several other papers [44–46]. This section will focus on putting together a solution based on previous work. The goal is to develop an algorithm that fits the requirements and problem description in this project.

7.4.1 Related work

Tor and Middleditch [44] defines an algorithm that traverses the edges of a polygon A. Each edge e_i is successively appended to either the current convex hull or one of the inner regions of the hull. The current hull H eventually grows into the final hull, where H = Hull(A). An edge is categorized based on its position relative to the current hull and the appropriate action is taken. The algorithm performs quadratic in the worst case and linear in the best case. It is expected that polygons in our case lie closer to the best case than the worst case regions discussed by Tor and Middleditch.

Jiao *et al.* [43] presents a greedy algorithm that divides a concave polygon into convex sub-regions. The algorithm chooses the solution that minimizes the sum of sub-regions' widths. Some of these sub-regions are combined to produce a more efficiently flight path. As shown in Figure 7.7, on the left hand side the path consists of one more edge than the combined version on the right.

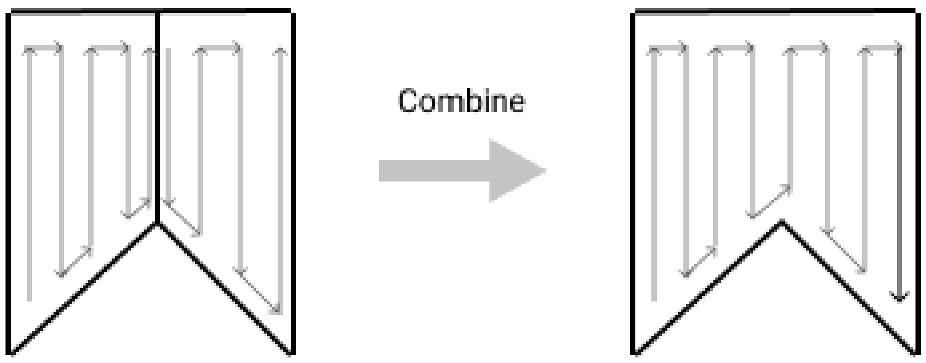


Figure 7.7: Fewer sub-regions are often better. We combine sub-regions to avoid redundant flight paths.

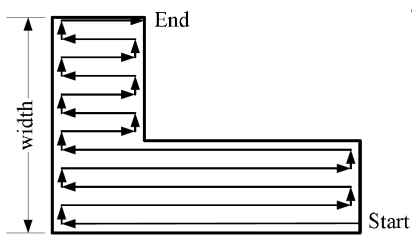
7.4.2 General Approach

The cover path planning algorithm proposed in this chapter follows a greedy approach, inspired by Jiao *et al.*'s work [43].

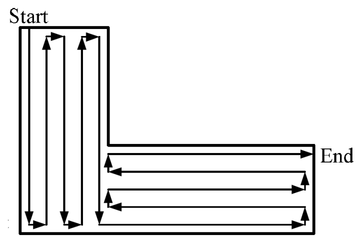
We proved in Section 7.3 that setting the sweeping direction parallel to the support lines minimized the number of turns. Figure 7.8a demonstrates how the previously defined coverage path approach produces a sub-optimal path on concave polygons. To deal with this, we propose a solution based on convex decomposition.

Jiao *et al.* propose a simple, yet effective algorithm for convex decomposition. The authors use a greedy recursive method that divides concave polygons into two sub-polygons by minimizing the sum of their widths. The basic idea behind the greedy technique is to always choose the solution that seems best at the moment [47].

To choose the optimal solution to a given problem, we need a comparison function. This function takes two solutions as input, compare them to each other, and greedy chooses and outputs the best solution. We let *a problem* be defined as a polygon with at least one concave vertex. A solution to this problem is a decomposition that divides the concave polygon into two smaller (possibly concave)



(a) Sub-optimal path.



(b) Optimal path.

Figure 7.8: Sub-optimal coverage path with sweep direction based on support lines (left) compared with an optimal coverage path (right).

polygons. We assume that each polygon produced by a decomposition will be a *simpler problem* to solve than the former.

So, how do we divide a polygon into two simpler polygons? For a starter, we know the following:

1. A concave polygon is defined as a polygon containing at least one concave vertex. In other words, at least one of the polygon's vertices is concave.
2. To reduce the concavity of a vertex, a new edge has to be drawn from that edge.

From 2 we need to determine in what direction the new edge are to be drawn. Jiao *et al.* shows that, using the polygon's width as a measure, the decomposition line is always parallel to one edge of the polygon when the sum of widths are minimized. For each concave vertex, the algorithm will have to consider all edges on that polygon and calculate the sum of widths. The minimum width sum are considered the optimal solution for that particular polygon.

7.4.3 Implementation

Her kommer det litt om implementasjonen...

7.5 Altitude

While the path algorithm in the previous section covers a 2D area efficiently it does not address the position of the aircraft in relation to the terrain. As the flight path define large areas, the altitude of points along the path will differ significantly. Therefore there are two problems that needs to be addressed in regards to the terrain in an area.

1. Obstacles due to difference in altitude. See figure 7.9b for an illustration highlighting this problem.
2. Field of view of the camera due to difference in altitude. See figure 7.9c for an illustration highlighting this problem.

With regards to obstacles, this mostly involves an increase in the terrain leading to a hill or mountain getting in the way of the drones flight path. When it comes to the field of view of the camera pictures should be taken at an optimal altitude to ensure the balance between a large area covered and the detail of the image. These two problems means that the final path needs to consider the altitude of each point in the path, *as well as points between*.

The algorithm defines a path consisting of straight lines between points points. As the terrain may differ significantly along a line between two points the altitude of intermediate points at a defined interval along the line needs to be added as a consideration. Figure 7.9a shows such a terrain difference. While the blue and red line go to the same point, the red line follows the terrain better due to having an intermediate altitude.

7.5.1 Height data in Norway

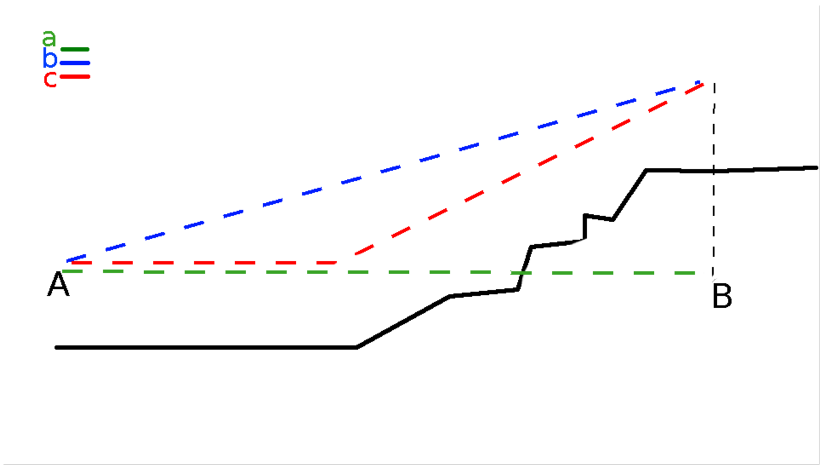
As altitude needs to be considered when calculating the final path, the system should have access to height data of the area, and use the data to add altitude parameters to the path defined by the path planning algorithm. This section shows the different height maps available in Norway and how they are added to the system. There are several height maps available in Norway. These depict the terrain in meters above sea level. They differ in accuracy and method of collecting, as well as some being a collection of multiple other data. All data is publicly available at *hoydedata.no*. [31] The data can be categorized into two: NN2000 and local point clouds of measurements.

NN2000

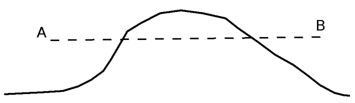
NN2000 is the current standard for altitude measurements in Norway. [32] It started in 2011 and was finished in 2018. It replaced NN1954, due to changes in the height of Norway as a whole. Norway rises slightly. NN2000 although started in 2011 uses the heights of the country in 2000. Later land rising is modelled and applied to these data. As such NN2000 should be accurate for the foreseeable future.

Local point clouds

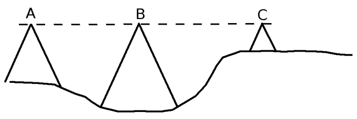
Local point clouds are part of the *Nation Detailed Height Model (NDH)*. [33] Point clouds are models of points in a 3D-space. In relation to height data, point clouds are detailed height measurements of the terrain in an area. NDH as a project aims to create a highly detailed model of Norway covering 253 000 km^2. It uses measurements from planes or helicopters with mounted laser scanners. The goal is an accuracy of 1mx1m. The NDH is a collection of local measurements. The data is available, but due to the size of the data it has to be downloaded as separate files for each local region.



(a) Shows different paths from a point 1 to point B. Green line disregards altitude completely. Blue line goes directly from point A to point B with the proper altitude increase over point B. The red line uses a intermediate point to keep a path that deviates minimal from the altitude relative to the terrain



(b) Shows the issue of disregarding terrain in regards to hitting objects. Also shows the need for intermediate points.



(c) Shows the issue in differing altitudes with regards to the area covered by the camera. Point A shows the ideal height. In point B details are lost in the picture due to the height, while point C inefficiently covers a relatively small area.

Figure 7.9: Illustrations regarding difference in altitude of the terrain

Conclusion

’ While the system could use both data sets, the ease of integrating NN2000, and due to currently being complete for the entirety of Norway, was chosen for this project. The NDH is not complete for Norway yet. The system in this project should be usable in most of Norway, especially in areas where animals graze, and NDH’s coverage in these areas is still lacking. As the drone will fly high above the ground, the accuracy of the height data in NN2000 does not need to be within 1 meters and NN2000 should be sufficient. In addition most of the data in NN2000 uses the same data as in NDH, meaning there are no difference for certain areas.

7.5.2 Coordinates

The algorithm used to define a path for covering an area is given coordinates in the form of longitudes and latitudes, and return a path with the same type of coordinates. However as the algorithm gives points for the drone to follow and the earth being a globe some challenges arise.

Great circle

Great circle in regards to navigation is the practice of using the shortest line between two given points.[34] The line follows the globe around it's curvature and end up in the same point. Any two points on the earths surface have an unique great circle, except points being directly on opposite sides of the globe, they will have an infinite amount of great circles. Using the great circle one can easily find intermediate points between two other points. Using Great Circle navigation the system ensures the drone takes the shortest path possible between points, while still following the curvature of the earth. Figure 7.10 illustrates a great circle derived from two points P and Q.

7.6 Camera

The altitude relative to the terrain of which the aircraft should fly depends on the purpose of the area coverage. If the purpose is spraying pesticides on a field, the height of the aircraft will be lower than if the purpose is to create a birds-eye view map of the area. As this project aims to create a solution for finding grazing animals the height at which the aircraft should fly above the terrain depends on the camera mounted on the drone. Aspects of the camera needs to be taken into consideration such as field of view (FOV), sensor, aperture and focal length.

7.6.1 Aspects of Cameras

As previously stated there are several aspects of cameras that need to be taken into consideration when determining the altitude of the aircraft. These aspects determine how well things are in focus, the detail of the image, how much of a scene is in the image and more. As such each relevant aspect is explained in this section, as well as how it affects the altitude of which the aircraft should travel.

Aperture

Aperture is the the opening on a camera lens where light passes through.[36] Aperture affects two things in an image, *brightness (or exposure)* and *depth of field*. The brightness of the image increases with a large aperture, and decreases with a small one. It is comparable with how the human eye works. In a bright room the pupils constrict to minimize the light hitting the retina, while the opposite is true for a dark room, where the pupils dilate to let more light in. As such one

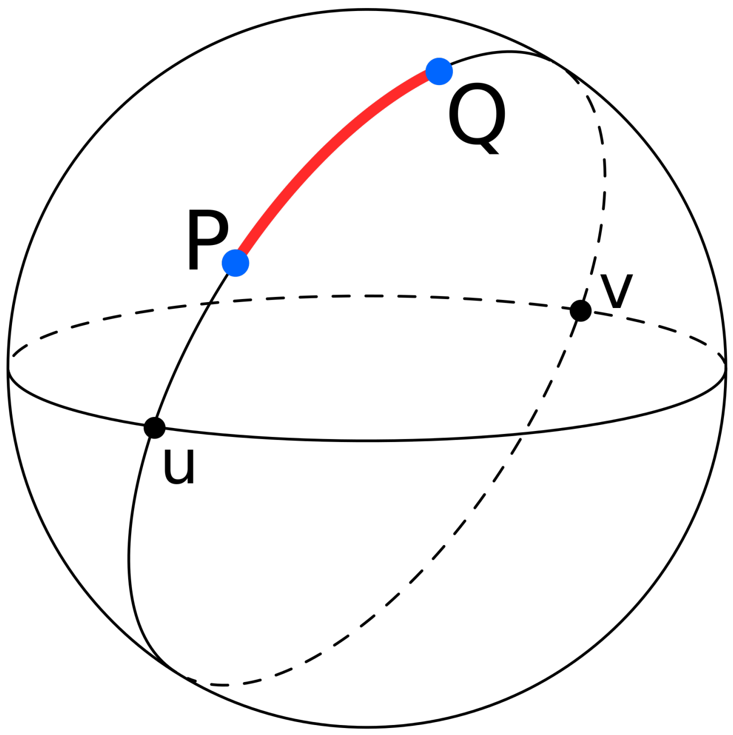


Figure 7.10: Illustration of a great circle line made from points P and Q. *"Illustration of great-circle distance" by CheCheDaWaff, licence: CC BY-SA [35]. No changes made.* `https://upload.wikimedia.org/wikipedia/commons/c/cb/Illustration_of_great-circle_distance.svg`

would want a large aperture if the environment is dark and a small aperture if the environment is bright.

The other thing aperture affects is the depth of field. With a large aperture, both the foreground and the background gets blurred relative to the focus point. With a small aperture the image is mostly clear both far away and close.

Aperture is depicted as f/x, where x is the inverse size of the aperture. As such a small x, such as f/1.4 means the aperture is large and the image is brighter and more unfocused in the foreground and background. While an aperture of f/16 means a small aperture that is less bright, but has a clear foreground and background.

Aperture needs to be considered as the depth of view determines how detailed objects of varying distance will be in the image, as well as the lighting of the environment being a factor.

Focus

Focus is directly correlated to the depth of field. At the *plane of focus* an object is sharp and detailed.[37] The plane of focus is the distance where the objects are at optimal sharpness. Objects closer to, and further away from the the plane of focus get exponentially more blurry. Most lenses can adjust the focus so that the desired object is in the range where the image is still sharp. Modern cameras has auto-focus capabilities.

Sensor

The sensor of a digital camera is what actually captures the image.[38] It is a chip consisting of millions of light sensitive elements called pixels. The light sensitive elements translates the incoming light into digital values. Sensors differ in several aspect, but the two most important to this project, are sensor size and amount of pixels.

The sensor size determines the actual size of the sensor. The standard size is 36x24mm. With the same lens a smaller sensor will show a subset of the image of a larger sensor, this is because the lens will focus the light around the smaller sensor, where a larger sensor will cover more of the light. As such the lens used on a camera will have different properties depending on the sensor size. A smaller sensor will have a smaller field of view with the same lens as a larger sensor.

The amount of pixels determines the resolution of the resulting image. As each pixel is converted to a digital value more pixels means images with a higher level of detail. The amount of pixels are usually denoted as megapixels, meaning the current cameras has a magnitude of millions of pixels.

The sensor is relevant to this project because it determines how much information the image holds, as well as the field of view.

Field and Angle of view

The *field of view (FOV)* is the section of the world observed from the camera.[39] *Angle of view(AOV)* is the angle determining the size of the field of view. A small angle of view, creates a narrow "cone" of what will be seen, while a large angle of view means more of the scene is covered in the image. Figure 7.11 shows how the angle and distance determines the size of what is covered in the scene and the difference in size depending on the distance. The angle of view depends on the sensor size and the focal length of the lens on the camera.

In photography, angle of view and field of view are often interchangeable, where field of view is the most common used and is given as an angle. In this paper for the sake of clarity, AOV is the angle, and FOV is the resulting coverage of the scene.

As images are mostly rectangular, the angle of view is different horizontally and vertically. As such it is common to describe the different values of the angle of view as *horizontal angle of view* (HAOV) and *vertical angle of view* (VAOV), and *diagonal angle of view*. The same for the different values of the field of view, respectively: *horizontal field of view* (HFOV) and *vertical field of view* (VFOV), and *diagonal angle of view* (DFOV).

To calculate the angle of view the focal length of the lens and the sensor size is needed. The formula 7.4 calculates the angle of view where, s is the sensor size, f is the focal length, and θ_{AOV} is the angle of view.

$$2arctan(\frac{s}{2f}) = \theta_{AOV} \tag{7.4}$$

Given the angle of view and a distance, the coverage of the resulting image can be calculated. The formula for finding the field of view given the angle of view is shown in 7.5, where θ_{AOV} is the angle of view of the camera, l is the distance to the scene, and FOV is the resulting field of view.

$$2tan(\frac{\theta_{AOV}}{2}) * l = FOV \tag{7.5}$$

It is also relevant to calculate the angle of view of one direction, given the angle of view of the other direction and the ratio of the sensor. Formula X and X show how to convert between HAOV and VAOV, where θ_{HAOV} is the horizontal angle of view, θ_{VAOV} is the vertical angle of view, and R is the ratio $(\frac{H}{V})$ of the sensor.

$$2arctan(\frac{tan(\frac{\theta_{HAOV}}{2})}{R}) = \theta_{VAOV} \tag{7.6}$$

$$2arctan(tan(\frac{\theta_{VAOV}}{2}) * R) = \theta_{HAOV} \tag{7.7}$$

To find the diagonal angle of view we first need to find the length of the diagonal on the sensor. The relationship between the diagonal and the length of each side is shown in formula 7.8

$$h^2 + v^2 = d^2 \tag{7.8}$$

Specifications

The camera specifications of the DJI Mavic 2 is shows in table 7.1 and are taken directly from the official manual REF. The specifications are both for the thermal and the visual camera. Some things to consider when looking at the specifications for the thermal camera are:

- The sensor size and lens is not given. However the sensor resolution is given.
- The the HFOV is given as an angle, this is the angle of view as defined in section 7.6.1. Horizontal direction is also the only one given.

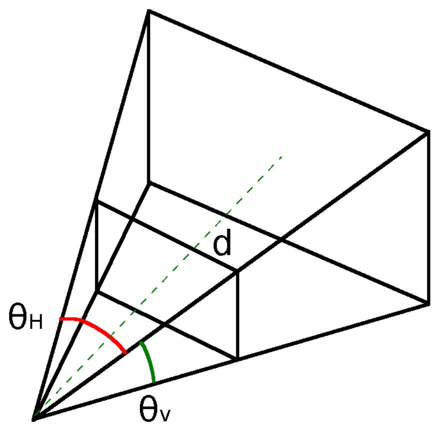


Figure 7.11: Figure showing how the angle of view and distance determine the area covered by an image

We can use the sensor resolution, and the horizontal angle of view to find out the ratio and from this calculate the vertical angle of view of the thermal camera

We get more information on the visual camera. Some things to consider when looking at the specifications for the visual camera are:

- 1/2.3" CMOS has a ratio of 4:3 REF.
- The effective pixels of 12M means there are 4056×3040 pixels REF.
- The direction of the FOV (angle of view) is not given.
- "35 mm format equivalent: 24 mm" means that the lens has the same focal length as a 24 mm lens on a 36x24 mm sensor.

Since the angle of view does not have a direction it will need to be calculated using the focal length and sensor size.

The lens is not able to zoom. The specifications are sufficient to calculate the area covered given a specified height.

Calculations

For this project, the coverage of an image from a given height is needed. The angle of view in both the horizontal and vertical direction can be used, with the height, to calculate the coverage. Using the specifications given in 7.1 we can use the formulas from 7.4-7.8 to find all relevant values. We need the AOV both horizontally and vertically for both the thermal and visual camera.

Starting with the thermal camera we already know the horizontal angle of view from table 7.1 as 57°. The sensor 160x120 meaning the ratio is 4/3. Using formula 7.6 we get a vertical angle of view of $\tilde{4}4.31°$.

Finding the horizontal and vertical angle of view of the visual camera is more difficult and some assumptions has to be made. The FOV from the manuals does not give a direction. Using formula 7.4 and a sensor of 36mmx24mm, results in a

Thermal camera	
Sensor	Uncooled VOx Microbolometer
Lens	HFOV: 57° Aperture: f/1.1
Sensor resolution	160x120
Visual camera	
Sensor	1/2.3" CMOS Effective pixels: 12M
Lens	FOV: approx. 85° 35 mm format equivalent:24 mm Aperture: f/2.8 Focus: 0.5 m to ∞

Table 7.1: DJI Mavic 2 Enterprise Dual camera specifications

diagonal of 84.11°. This is close to the approximation of 85°given in the manual and it can be assumed that this is the diagonal angle of view.

The problem with this value is that it is a result of having a sensor with an aspect ratio of 3:2. The actual sensor has a ratio of 4:3. The most likely reason for this discrepancy is that all the information on the lens follows the 35 mm equivalent and this has a sensor ratio of 3:2.

To convert the angle of view given a 3:2 ratio to the horizontal and vertical angle of view given a 4:3 format we can change the sensor size of the 35mm equivalent to 36mmx27mm, which has a ratio of 4:3, and use this with formula 7.4. Using a f of 24mm and s as the sensor size for a given direction we get:

- Finding the vertical field of view:
  $2arctan(\frac{27mm}{2*24mm}) = 51.72°$
- Finding the horizontal field of view:
  $2arctan(\frac{36mm}{2*24mm}) = 73.74°$

As we now have the angle of views of both the thermal and visual camera we can calculate the field of view in both directions given a height.

Implementation

As the different angle of views of both the thermal and visual camera is known they can be used to find the field of view. Code listing 7.1 shows the implementation for finding the field of view. Formula 7.5 was translated to Javascript code and can be found on line 6 in code listing 7.1. Using function *getFov* a object with all field of views are returned for a provided height. Table 7.2 shows the different field of views given any height.

```
1 const degreesToRadians = (degrees) = {
   return degrees * (Math.PI/180);
};

const fov = (angleOfView, height) => {
   return 2 * Math.tan(degreesToRadians(angleOfView) / 2) * height;
};

const aov = {
   // Thermal
   thermal: {
      horizontal: '57',
      vertical: '44.31'
   },
   // Visual
   visual: {
      horizontal: '74.74',
      vertical: '51.72'
   }
};

const getFov = (height) => {
   return {
      thermal: {
         horizontal: fov(aov.thermal.horizontal, height),
         vertical: fov(aov.thermal.vertical, height)
      },
      visual: {
         horizontal: fov(aov.visual.horizontal, height),
         vertical: fov(aov.visual.vertical, height)
      }
   };
}
```

Code listing 7.1: Javascript implementation for finding the field of view of the drone given a height

Height / **Camera**	10 m	50 m	100 m
Thermal	10.86x8.14 m	54.30x40.72 m	108.6x81.4 m
Visual	15.27x9.69 m	76.37x48.47 m	152.7x96.9 m

Table 7.2: Shows field of views of the thermal and visual camera at different heights

Conclusion

By looking at the components of a camera, like the sensor and lens, properties of the camera can be found. Formulas for finding the angle and field of view were used with the camera specifications from the DJI Mavic 2 Enterprise. After finding the angle of view of both the thermal and visual camera, a simple Javascript program was implemented that can be used to find the field of view of the camera on the drone given a particular height.

Bibliography

[1] M. Silvagni, A. Tonoli, E. Zenerino and M. Chiaberge, ‘Multipurpose uav for search and rescue operations in mountain avalanche events’, *Geomatics, Natural Hazards and Risk*, vol. 8, no. 1, pp. 18–33, 2017.

[2] L. F. Gonzalez, G. A. Montes, E. Puig, S. Johnson, K. Mengersen and K. J. Gaston, ‘Unmanned aerial vehicles (uavs) and artificial intelligence revolutionizing wildlife monitoring and conservation’, *Sensors*, vol. 16, no. 1, p. 97, 2016.

[3] V. Gatteschi, F. Lamberti, G. Paravati, A. Sanna, C. Demartini, A. Lisanti and G. Venezia, ‘New frontiers of delivery services using drones: A prototype system exploiting a quadcopter for autonomous drug shipments’, in *2015 IEEE 39th Annual Computer Software and Applications Conference*, IEEE, vol. 2, 2015, pp. 920–927.

[4] M. of Agriculture and Food. (2010). Lov om dyrevelferd, [Online]. Available: `https://lovdata.no/dokument/NL/lov/2009-06-19-97`.

[5] M. of Agriculture and Food. (2008). Forskrift om velferd for småfe, [Online]. Available: `https://lovdata.no/dokument/SF/forskrift/2005-02-18-160`.

[6] S. Norway. (2020). Livestock grazing on outfield pastures, by region, contents and year, [Online]. Available: `https://www.ssb.no/en/statbank/table/12660/chartViewColumn/`.

[7] H. A. Solbakken and S. H. Berge. (2019). Nedgang i tap av sau til ulv på beite i 2019, [Online]. Available: `https://www.nrk.no/innlandet/markant-nedgang-i-tap-av-sau-til-ulv-pa-beite-i-2019-1.14661400`.

[8] S. Tallaksrud. (2017). Ulven er ikke sauens største fiende, [Online]. Available: `https://www.nrk.no/innlandet/xl/ulven-er-ikke-sauens-storste-fiende-1.13566997`.

[9] J. Linnestad and O. H. Ødegaard, ‘Manuell oppfølging av sau på beite’, Master’s thesis, Norwegian University of Science and Technology, Dec. 2019.

[10] C.-M. Tseng, C.-K. Chau, K. M. Elbassioni and M. Khonji, ‘Flight tour planning with recharging optimization for battery-operated autonomous drones’, *CoRR, abs/1703.10049*, 2017.

[11] C. Clips. (Nov. 2019). Territorial bird attacks flying drone, [Online]. Available: https://www.youtube.com/watch?v=1NY3Df_Wuc4.

[12] C. Clips. (Nov. 2015). Phantom 3 get kidnapped by two eagles, [Online]. Available: https://www.youtube.com/watch?v=FX3uOQiZs0A.

[13] Luftfartstilsynet. (2020). Luftfartstilsynet (civil aviation authority), [Online]. Available: https://luftfartstilsynet.no/.

[14] Stortinget (Supreme Legislature of Norway), Ministry of Transport, *Forskrift om luftfartøy som ikke har fører om bord mv.* 30th Nov. 2015. [Online]. Available: https://lovdata.no/dokument/SF/forskrift/2015-11-30-1404.

[15] H. U.S. Department of Defense (DoD) General Atomics Corp. and Raytheon. (2020). Mq–1b predator / mq-1c gray eagle / mq–9 reaper, [Online]. Available: http://www.fi-aeroweb.com/Defense/MQ-1-Predator-MQ-9-Reaper.html.

[16] JoJo. (May 2020). Types of drones – explore the different models of uav's, [Online]. Available: http://www.circuitstoday.com/types-of-drones.

[17] M. Borak. (2018). World's top drone seller dji made $2.7 billion in 2017, [Online]. Available: https://technode.com/2018/01/03/worlds-top-drone-seller-dji-made-2-7-billion-2017/.

[18] DJI. (Sep. 2019). Mavic 2 enterprise series - user manual, [Online]. Available: https://dl.djicdn.com/downloads/Mavic_2_Enterprise/20190917/Mavic_2_Enterprise_Series_User_Manual-EN.pdf.

[19] IDC. (Feb. 2020). Smartphone market share, [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os.

[20] SSB. (2018). Norsk mediebarometer, [Online]. Available: https://www.ssb.no/statbank/table/05244/tableViewLayout1/.

[21] G. Cai, B. M. Chen and T. H. Lee, 'Coordinate systems and transformations', in *Unmanned rotorcraft systems*, Springer, 2011, p. 26.

[22] yr.no. (2020). Yr - 'trondheim' - statistics, [Online]. Available: https://www.yr.no/en/statistics/graph/1-211102/Norway/Tr%C3%B8ndelag/Trondheim/Trondheim?q=2020-03-10.

[23] A. Inc. (2020). Human interface guidelines - design - apple developer, [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/.

[24] A. Inc. (2020). App and website maps - maps - human interface guidelines - apple developer, [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/maps/overview/introduction/.

[25] G. LLC. (2020). Google maps, [Online]. Available: https://www.google.com/maps.

[26] A. Inc. (2020). Apple maps, [Online]. Available: `https://www.apple.com/ca/ios/maps/`.

[27] Mapquest. (2020). Mapquest - maps, driving directions, live traffic, [Online]. Available: `https://www.mapquest.com/`.

[28] O. S. M. (source). (2020). Mapquest - maps, driving directions, live traffic, [Online]. Available: `https://www.openstreetmap.org/about`.

[29] M. (source). (2020). Maps.me (built on open street map), [Online]. Available: `https://maps.me`.

[30] J. S. Dumas, J. S. Dumas and J. Redish, *A practical guide to usability testing*. Intellect books, 1999, p. 259.

[31] (Apr. 2020). Høydedata, [Online]. Available: `https://hoydedata.no/LaserInnsyn/`.

[32] (Apr. 2020). Nn2000, [Online]. Available: `https://www.kartverket.no/NN2000`.

[33] (Apr. 2020). Om nasjonal detaljert høydemodell (ndh), [Online]. Available: `https://www.kartverket.no/Prosjekter/Nasjonal-detaljert-hoydemodell/om-nasjonal-detaljert-hoydemodell/`.

[34] (Apr. 2020). Great circle, [Online]. Available: `https://www.naturalnavigator.com/news/2018/12/what-is-a-great-circle/`.

[35] *Attribution-sharealike 4.0 international (cc by-sa 4.0)*, `https://creativecommons.org/licenses/by-sa/4.0/`, Apr. 2020.

[36] N. Mansurov. (Apr. 2020). Understanding aperture in photography, [Online]. Available: `https://photographylife.com/what-is-aperture-in-photography`.

[37] S. Cox. (Aug. 2019). Understanding focus in photography, [Online]. Available: `https://photographylife.com/understanding-focus-in-photography`.

[38] M. Golowczynski. (Jun. 2016). Digital camera sensors explained, [Online]. Available: `https://www.whatdigitalcamera.com/technical-guides/technology-guides/sensors-explained-11457`.

[39] D. Carr. (Jun. 2017). Angle of view vs. field of view. is there a difference and does it even matter?, [Online]. Available: `https://shuttermuse.com/angle-of-view-vs-field-of-view-fov-aov/`.

[40] H. L. Andersen, 'Path planning for search and rescue mission using multicopters', Master's thesis, NTNU, Institutt for teknisk kybernetikk, 2014.

[41] M. Coombes, W.-H. Chen and C. Liu, 'Boustrophedon coverage path planning for uav aerial surveys in wind', in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2017, pp. 1563–1571.

[42] C. Di Franco and G. Buttazzo, 'Coverage path planning for uavs photogrammetry with energy and resolution constraints', *Journal of Intelligent & Robotic Systems*, vol. 83, no. 3-4, pp. 445–462, 2016.

[43] Y.-S. Jiao, X.-M. Wang, H. Chen and Y. Li, 'Research on the coverage path planning of uavs for polygon areas', in *2010 5th IEEE Conference on Industrial Electronics and Applications*, IEEE, 2010, pp. 1467–1472.

[44] S. B. Tor and A. E. Middleditch, 'Convex decomposition of simple polygons', *ACM Transactions on Graphics (TOG)*, vol. 3, no. 4, pp. 244–265, 1984.

[45] J.-M. Lien and N. M. Amato, 'Approximate convex decomposition of polygons', in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 17–26.

[46] J. Li, W. Wang and E. Wu, 'Point-in-polygon tests by convex decomposition', *Computers & Graphics*, vol. 31, no. 4, pp. 636–648, 2007.

[47] (Apr. 2020). Basics of greedy algorithms, [Online]. Available: `https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/tutorial/`.

[48] K. K. Landes, 'A scrutiny of the abstract', *Bulletin of the American Association of Petroleum Geologists*, vol. 35, no. 7, p. 1660, 1951.

Paper I

Here, you may add a description of the paper, an illustration, or just give the bibliographic reference:

> K. K. Landes, ‘A scrutiny of the abstract’, *Bulletin of the American Association of Petroleum Geologists*, vol. 35, no. 7, p. 1660, 1951

Or you may leave it empty, if you like.

BULLETIN OF THE AMERICAN ASSOCIATION OF PETROLEUM GEOLOGISTS
VOL. 50, NO. 9 (SEPTEMBER, 1966), P. 1992-1999, 2 FIGS.

GEOLOGICAL NOTES

A SCRUTINY OF THE ABSTRACT, II[1]

KENNETH K. LANDES[2]
Ann Arbor, Michigan

ABSTRACT

A partial biography of the writer is given. The inadequate abstract is discussed. What should be covered by an abstract is considered. The importance of the abstract is described. Dictionary definitions of "abstract" are quoted. At the conclusion a revised abstract is presented.

For many years I have been annoyed by the inadequate abstract. This became acute while I was serving a term as editor of the *Bulletin* of The American Association of Petroleum Geologists. In addition to returning manuscripts to authors for rewriting of abstracts, I also took 30 minutes in which to lower my ire by writing, "A Scrutiny of the Abstract."[1] This little squib has had a fantastic distribution. If only one of my scientific outpourings would do as well! Now the editorial board of the Association has requested a revision. This is it.

The inadequate abstract is illustrated at the top of the page. The passive voice is positively screaming at the reader! It is an outline, with each item in the outline expanded into a sentence. The reader is told what the paper is about, but not what it contributes. Such abstracts are merely overgrown titles. They are produced by writers who are either (1) beginners, (2) lazy, or (3) have not written the paper yet.

To many writers the preparation of an abstract is an unwanted chore required at the last minute by an editor or insisted upon even before the paper has been written by a deadline-bedeviled program chairman. However, in terms of market reached, the abstract is *the most important part of the paper.* For every individual who reads or listens to your entire paper, from 10 to 500 will read the abstract.

If you are presenting a paper before a learned society, the abstract alone may appear in a preconvention issue of the society journal as well as in the convention program; it may also be run by trade journals. The abstract which accompanies a published paper will most certainly reappear in abstract journals in various languages, and perhaps in company internal circulars as well. It is much better to please than to antagonize this great audience. Papers written for oral presentation should be *completed prior to the deadline for the abstract,* so that the abstract can be prepared from the written paper and not from raw ideas gestating in the writer's mind.

My dictionary describes an abstract as "a summary of a statement, document, speech, etc. . . ." and that which *concentrates in itself the essential information* of a paper or article. The definition I prefer has been set in italics. May all writers learn the art (it is not easy) of preparing an abstract containing the *essential information* in their compositions. With this goal in mind, I append an abstract that should be an improvement over the one appearing at the beginning of this discussion.

ABSTRACT

The abstract is of utmost importance, for it is read by 10 to 500 times more people than hear or read the entire article. It should not be a mere recital of the subjects covered. Expressions such as "is discussed" and "is described" should *never* be included! The abstract should be a condensation and concentration of the *essential information* in the paper.

[1] Revised from K. K. Landes' "A Scrutiny of the Abstract," first published in the *Bulletin* in 1951 (*Bulletin,* v. 35, no. 7, p. 1660). Manuscript received, June 3, 1966; accepted, June 10, 1966.

Editor's note: this abstract is published together with The Royal Society's "Guide for Preparation and Publication of Abstracts" to give *Bulletin* authors two viewpoints on the writing of abstracts.

[2] Professor of geology and mineralogy, University of Michigan. Past editor of the *Bulletin.*

Appendix A

Additional Material

Additional material that does not fit in the main thesis but may still be relevant to share, e.g., raw data from experiments and surveys, code listings, additional plots, pre-project reports, project agreements, contracts, logs etc., can be put in appendices. Simply issue the command `\appendix` in the main `.tex` file, and make one chapter per appendix.

If the appendix is in the form of a ready-made PDF file, it should be supported by a small descriptive text, and included using the `pdfpages` package. To illustrate how it works, a standard project agreement (for the IE faculty at NTNU in Gjøvik) is attached here. You would probably want the included PDF file to begin on an odd (right hand) page, which is achieved by using the `\cleardoublepage` command immediately before the `\includepdf[]{}` command. Use the option `[pages=-]` to include all pages of the PDF document, or, e.g., `[pages=2-4]` to include only the given page range.

NTNU

Vår dato Vår referanse

Norges teknisk-naturvitenskapelige universitet

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

____

____ (oppdragsgiver), og

____

____

____ (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra ____________ til______________ .

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

   Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

   Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): ____

Oppdragsgivers kontaktperson (navn): ____

Student(er) (signatur): ____ dato ___________

____ dato ___________

____ dato ___________

____ dato ___________

Oppdragsgiver (signatur):____ dato ___________

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): __________________________________ dato ___________