NTNU
Norwegian University of
Science and Technology

Pecora: Presentation of Free Range Sheep Information

Julie Johnsen Kirkhus

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Svein-Olaf Hvasshovd, IDI
Co-supervisor: Theoharis Theoharis, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

The main goal of this research is to help sheep farmers find and track their sheep, especially at the end of the grazing season, when the sheep must return to the farms. This research aims at delivering the presentation part of a IT-system named Pecora. Pecora will provide the sheep information by taking thermal images from a drone. The presentation part of Pecora is the interface between the farmers and the results provided by the drone. The presentation part is named Pecora/P. Pecora/P will deliver and display all the information the farmers need in order for them to find their sheep in the wild. This report is about the development and testing of Pecora/P. Pecora/P will be delivered as a webpage containing a map and an information panel. The information panel will display all the information available about each of the findings of sheep. The map will be used for displaying the sheep and other related animals found during the analysis of the pictures in an visual presentation. Sheep and other different animals will be placed on the map with specific icons, based on the species of the herd. The webpage will also provide to the users all the information about findings from previous days. This gives the farmers the ability to see the route the sheep have taken over the past week. Pecora/P are deliberately made modifiable as Pecora are not fully implemented, and Pecora/P are open to new approaches that can be used to find and track sheep during the grazing period.

Preface

This Master's Thesis was written during the spring 2016 at the Department of Computer and Information Science at Norwegian University of Science and Technology.

The Master's Thesis is written under the Software specialization, and will cover the design phase, the implementation phase and the testing phase of developing an information system.

The student would like to thank the project's supervisor Svein-Olaf Hvasshovd for his help, great advices and constructive feedback, and for the interesting discussions bringing the thesis forward. The student would also like to thank Frederik Stendahl Leira from the Department of Engineering Cybernetics for his contribution of the drone, and the farmers Hallvard Storli and Steingrim Horvli for giving us their time for testing the information system and giving constructive feedback on the usability.

24. June, 2016

Summary

Hovedmålet med denne forskningen er å hjelpe saubønder med å finne og spore sauene sine, spesielt i slutten av beitesesongen, da sauene må tilbake til gårdene. Denne forskningen tar sikte på å levere presentasjons delen av et IT-system navngitt Pecora. Pecora vil tilby information om sauene ved bruk av termiske bilder tatt av en drone. Presentasjons delen av Pecora er grensesnittet mellom brukerene (bøndene) og informationen om sauene hentet fra dronen. Presentasjons delen er navngitt Pecora/P.

Pecora/P vil levere og vise all informationen bøndene vil trenge for å finne saune deres uti naturen. Denne rapporten dreier seg om utvikling og testing av Pecora/P. Pecora/P vil bli levert som en nettside som vil inneholde et kart og et informasjonspanel. Informasjonspanelet vil vise all information tilgjengelig av hvert enkelt funn av sau. Kartet vil bli brukt for a vise sauene og andre relevante dyr som er funnet under bildeanalysen på en visuell metode. Sau og andre relevante dyr vil bli plassert på kartet med bruk av forskjellige ikoner for å vise hva slags type dyr som er funnet.

Nettsiden vil også tilby brukerne all information om sau og andre dyr funnet de foregående dagene. Dette gir bøndene muligheten til å se hvilken rute sauene kan ha fulgt de forrige ukene. Pecora/P er bevisst gjort modifiserbar siden Pecora er ikke utvikliet enda, og Pecora/P må være open til nye forandringer og måter å finne og spore sau på under beitesesongen.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This report deals with the farmers' problems with finding and gathering sheep in the wild, and how these challenges can be solved with the application of technology. In order to give a good introduction to the master thesis, this chapter will start by giving the background and motivation for why the project was initiated (Section 1.1), whereas the goals and research questions connected to the mater thesis are presented in Section 1.2. The method and approach for achieving the goals and answering the research questions are explained in Section 1.3. The last section, Section 1.4, will present the structure of the report.

1.1 Background and Motivation

Every year, about 2 million sheep are let out to graze in Norway (nø; NINA, 2014). In late September to early October are the sheep found and retrieved back to the farms. The farmers struggles with finding all the sheep, and they need help. With information technology and new technologies, this project will be the start and presentation part of a new information technology system named Pecora. Pecora will help the farmers with finding their sheep so that the farmers can bring them back to the farm. Pecora uses a drone to take pictures of the ground below over areas known to have sheep. The pictures will then be analysed to check if any sheep are in the pictures. The result will then be presented to the farmers. The presentation part of Pecora is called Pecora/P. This report will cover every aspect of the design phase, the implementation phase and testing phase of Pecora/P. Pecora/P will take the information gathered by the drone and the pictures and give the farmers a full overlook of where their sheep are located in the wild. Pecora/P are the interface between the user and Pecora.

1.2 Goals and Research Questions

This project will deliver a presentation (webpage) of data gathered by a drone and generated through image analysis, as part of the information technology system named Pecora.

During this project the presentation will be developed and tested by two local farmers. The focal question is how to present the free range sheep information. The presentation must be able to present the data in a way the users understand. The users need to use the information later in order to find their sheep. The presentation of the data must therefore be readable and understandable for the user. The main goal is therefore as follows:

Goal Develop a system that displays sheep herds and other related animals in an unambiguous manner along with information necessary for sheep-farmers to find their sheep.

The project will deliver answers to some preset research questions. The research questions focus on how to present information about sheep and related animals in an unambiguous manner. The research questions are related to the final delivery of this project, Pecora/P. The research questions are as follows:

RQ1 What is the best approach for displaying findings of sheep and related animals efficiently to farmers?

RQ2 What information is important to give to the farmers about the findings of sheep and related animals?

RQ3 How can a map display the findings of sheep and related animals in an unambiguous manner?

1.3 Research Method

The research method used to achieve the goals stated in the previous section is called Design Science Research. Design Science Research gives guidelines for executing research in information systems (DSR, 2016). The different guidelines can be seen in Figure 1.1. This project will focus on the design cycle. The design cycle refers to designing and building artifacts and features, then to run an evaluation once they are finished. Then a new design round is run, then evaluated. This continues until the program reaches its goal. During the project will one relevance cycle be run. The relevance cycle refers to field testing, which means testing the program on farmers that will end up using the program, in this case the sheep-farmers. The rigor cycles will be run when there is a need for more information about today's farmers and Pecora as a whole system.

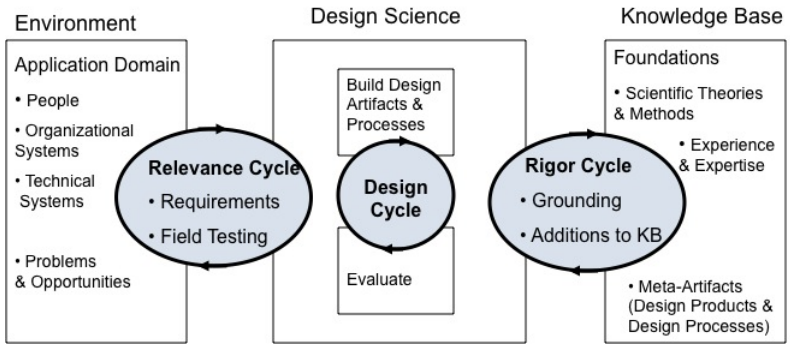

Figure 1.1: Design Science Research

1.4 Thesis Structure

The remaining part of the report is structured as follows. Chapter 2 - Background and Motivation will give a short background of today's situation in the sheep farming business and end with a full description of the whole system this project is a part of, which is called Pecora. After Pecora is described in full detail, the architecture of Pecora/P will be described in Chapter 3 - Architecture. Chapter 4 - Implementation will then cover the procedure and choices of technology used to implement Pecora/P. After implementing Pecora/P, a user-test was performed which is described in Chapter 5 - Testing. The results from the project and the fully implemented Pecora/P will be covered in Chapter 6 - Results. The last chapter, Chapter 7 - Discussion and Conclusion, will end this report with a conclusion followed by recommendations for future work.

Chapter 2

Background and Motivation

In Norway sheep are let out in the summer for grazing. The sheep are completely free to walk wherever they want during the grazing period. They are let out in the middle of May and have about 5-6 months where they can graze and move around. The end of the grazing period is around the last weeks of September to the middle of October, depending on the season and each individual farmer. During the end of the grazing period, all the sheep will be gathered and brought back to the farm. The total number of grazing sheep in Norway is about 2 million. This number is slowly increasing yearly (nø; NINA, 2014). Each farmer can have 600-700 sheep or more.

The gathering of sheep takes a lot of time and effort. The farmers will have to find the sheep in the wild, and during the grazing period the sheep herd will normally split into several smaller herds. The split herds may walk anywhere, and the farmers have few clues of where they may be. This means the farmers must walk in the woods and look for the sheep, and if they are lucky, they will meet a few. The farmers are of course familiar with specific areas where the sheep tend to go during the grazing period. They will find most of the sheep there by listening for the bells some sheep have around their neck. It is the herds that are in places where they usually do not go, or in areas where it is hard to hear the bells that are difficult to find. The sheep can be 20 meters away, and the farmers might not see or notice them. If the farmers do not know that the sheep are close by, they might walk away from the herd. The biggest challenge is that they do not know where to look for the sheep, as they can be anywhere. They need help with finding their whereabouts.

A new technology often used by farmers is to attach radios to their sheep. However radios are quite expensive to purchase. Some sheep also tend to tear the radio off during the grazing period. There are therefore only a small number of the sheep that have a radio attached (Jim Hultgreen, 2002; Anne Sigrid Haugset, 2010). However the farmers are recommend to use the radios as the sheep wearing a radio will be easier to find at the end of the grazing period. However, the terrain can cause the radio signal to be interrupted and the sheep may walk outside of radio reach. The radios work great in areas without tall

maintains and overgrown forests.

Thousands of sheep die every year due to attacks by predators or for walking into difficult and dangerous environments (Bondevennen.no, 2012; NINA, 2014, 2015). Predators such as lynx, wolverine, bear and even the golden eagle are know predators that have taken a liking to sheep. Every year the farmers are reimbursed for their missing sheep, even though it is not proven that the cause of death due to a predator (Larsen, 2011; NINA, 2014; Bondevennen.no, 2012). In 2011 were the farmers reimbursed for around 140 million NOK, that was 19 millions NOK more than the previous year, and the number is still increasing (Larsen, 2011). There are often discrepancies between the number of predators in the areas and the number of missing sheep (Larsen, 2011). The number of sheep killed by predators in Norway are notable higher than in Sweden, even though the number of predators in Sweden are higher than in Norway (Lie, 2012). This means that sheep die not only by predators but also from other causes as well. For example sheep can end up in dangerous environments causing sheep to fall down steep mountain sides or get stuck in the swamps. The lack of ability to track the sheep during the grazing period makes it difficult to know the cause of death. A dead sheep will end up as mere bones at the end of the grazing period, when the farmers look for them. The farmers sees them as lost, and/or eaten by predators.

How can technology help farmers find the whereabouts of their sheep, and how can the sheep be tracked during the grazing period. The next section describes a system that can help the farmers to find their sheep and help them track their whereabouts during the grazing period. A system named Pecora will use trending technologies such as a drone to take infrared and normal pictures, run a picture analysis and then display the findings of sheep and other related animals on a webpage using a map.

2.1 Pecora

Pecora is a three part system that will help farmers find their sheep preferably at the end of the grazing period. It will use drones and infrared cameras to find the missing sheep. The three main parts of Pecora are the drones (Subsection 2.1.1), the picture analysis (Subsection 2.1.2) and the presentation (Subsection 2.1.3).

Pecora uses a drone that can fly over predefined areas and take a series of pictures during the flight. The drone will calculate the coordinates and the time for each picture. The pictures will be taken by an infrared camera to start with and can be extended to use a regular camera together with the infrared camera. The infrared camera lets the picture analysis use the sheep's body temperatures to find them on the pictures. At the end of the grazing period, will the leaves on the trees and the bushes fall off and the infrared camera will be able to detect them. The infrared camera work best in the fall, as a hot summers day can warm the nature surrounding the sheep, and the surroundings can become equally warm as the sheep. In the fall however the surroundings will be cold and the sheep will be easier to spot in the pictures.

When the drone lands the pictures will be sent to the picture analysis part of the system. There the pictures will be run through an algorithm to determined if there are any animals in the picture. If there are no signs of animals in the picture, the picture is discarded. The rest will then be run through a new algorithm that will try to determine if the animals are sheep or some other species. The result will then be presented to the users. The presentation part of Pecora is called Pecora/P.

2.1.1 The Drone

The tasks of the drones in Pecora is to fly anywhere the users suspect there might be some missing sheep. It then has to take pictures rapidly while it flies over the predefined areas. It would be a great benefit if the drone to send pictures down to the farm during the flight. This is hard to implement, so for the time being the drone will wait until it has landed to offload the pictures. Along with the pictures, the drone will send the time they were taken as well as the coordinates, to the next part of Pecora; the picture analysis.

Pecora has been provided to test with two drones by the Department of Engineering Cybernetics. The team behind Pecora have been in contact with Frederik Stendahl Leira, a PHD candidate at the Department of Engineering Cybernetics. The two drones are installed with an infrared camera and both can calculate the coordinates in latitude and longitude and holds track of the time. Both drones can install regular cameras as well, though they will not be included in the early testing phase of Pecora. The smallest drone will be used in the early testing phase. The smallest of the drones is called Skywalker and runs on battery, which gives the Skywalker a speed up to 17 meters per second and it can fly up to 45 minutes to one hour. The Skywalker can fly 120 meters above the ground or higher, though that requires the users to apply for a permit. The camera attached can take a picture every 7.5 seconds. The largest of the drones is called Cloud Cap and are created by Uav Factory. The goal is to use this drone for Pecora. It runs on gasoline and can therefore fly up to eight-nine hours as opposed to Skywalkers which can fly max one hour. That will let Pecora cover a larger area and more sheep can be found in one run. As Pecora is only in the beginning of development, the Skywalker have been used to achieve the test data. The Skywalker can be seen in Figure 2.1 and the Cloud Cap can be seen in Figure 2.2.

Figure 2.1: The Skywalker

Figure 2.2: The Cloud Cap

The pictures above show the two drones Pecora is expected to use. The first drone is the black Skywalker, which are the first drone to be tested with Pecora. The second drone is the Cloud Cap, and is the drone which hopefully Pecora will use in the end. The Cloud

Cap is shown without its full wings attached.

The Flight Path

The drones will follow a specific path on each of their flights. The drones will fly back and forth covering a rectangular area, like a lawn mower. This path will make sure there is little overlap between the pictures. It will also ensure that a certain area has been fully covered. The flight path can be seen in Figure 2.3.

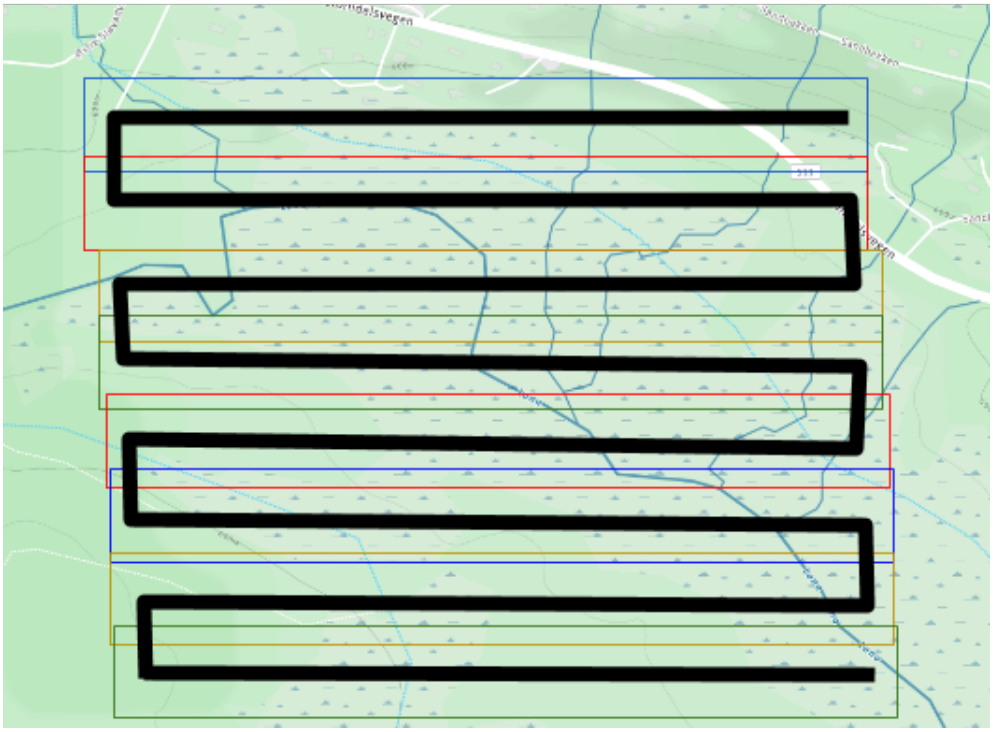

Figure 2.3: The Flight Path

The flight path is shown as a black line which goes back and forth. The colored rectangles show the overlaps in the pictures. The pictures cover a larger area than the drone itself, and some overlap in the pictures will therefore most certainly occur. The colors are used to illustrate the overlap in an more ambiguous manner. The overlap between the pictures are useful when covering a large area. Without the overlap, some sheep could be between pictures and therefore not spotted. The area will be finished for further search, at least for some days, and the sheep between the pictures would not be found. It is therefore crucial that the pictures have some overlap, to be sure the area is fully searched.

2.1.2 The Picture Analysis

The picture analysis in Pecora takes the pictures collected by the drones and uses an algorithm to locate sheep and other related animals. The algorithm will try to find the animals in the pictures by using their body temperature. The pictures will be brighter where the animals are captured, and the rest should be blackened, especially in the fall and winter. Warmer temperatures will be brighter than cold temperatures, and animals are warmer than the surroundings. If there is an animal in the picture, then the picture analysis will determine whether it is a sheep or another species by looking at the animals body ratios. The length of the animal combined with the width could tell if it is a sheep or a different animal. The picture analysis should be able to tell how many animals there are in one herd. If possible, the analysis will try to tell whether a sheep is a lamb or not, even though this might be hard to determine at the end of the grazing period. Lambs will grown during the grazing period, and at the end, lambs and adult sheep will be similar in size, and it is therefore hard to see the difference. Predators can be detected as well and the picture analysis should send a danger signal to Pecora/P in case a predator is spotted.

To use the size of the animals to determine which species they are, the size of the area covered by one picture must be calculated. Then the ratio between the animals- and picture area can tell how big the animals really are. The infrared camera installed into the Skywalker have predefined angles between the camera and the ground below. Figure 2.4 shows the angles between camera and ground seen from the front and the side of the drone.

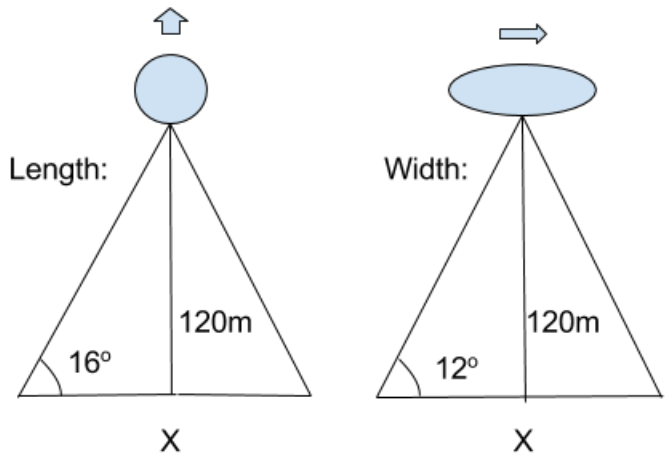

Figure 2.4: The Area Covered by One Picture

The figure to the left shows how much the camera captures on the sides. The angle between the camera and the sides are 16°. The figure to the right shows how much the camera captures in the front and back of the drone. The angle between the camera and the front

and back are 12°. Given that the drone flies at a height of 120 meters, the area covered by one picture can be calculated using the tangent. That gives the infrared camera installed on the Skywalker an ability to capture an area that is 70 meters long and 50 meters wide in every single picture.

Herds in the Pictures

The large areas covered by one picture makes it possible that several herds of animals are found in the same picture. After the picture analysis is done, the result will be sent to the presentation (Pecora/P). Each herd will be given its own spot in the presentation. Instead of adding all the herds found in one picture into one spot, they are divided as they are in the original picture. That means that each little group of sheep will have its own spot. If the picture contains two animal groups off different species, they are divided instead of combined. The area covered in one picture makes it to large to combine the herds. The herds can be up to 70 meters apart in the same picture. That would not be a good indicator of where the sheep herds are, as there can be two herds in the corners of the picture, but the farmers will get the information about one herd in the center of the picture. Each herd will be given a set of coordinates representing the center of the herd. In that way the users can see the center of a herd, instead of each animal having its own coordinates. Otherwise it would be to much information to give the farmers, and all that information is not necessary in order to locate the sheep in the wild afterwards.

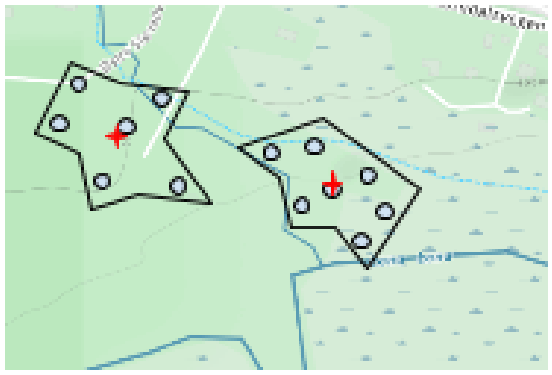

Figure 2.5: Two Sheep Herds

The figure above shows how two herds have been split and given a center. These two herds will be sent as separate spots to Pecora/P, and will be placed based on their center coordinates. Because of the flight path there will be some overlap in the pictures. The Figure 2.6 shows how four pictures have located several herds, and how some of them overlap.

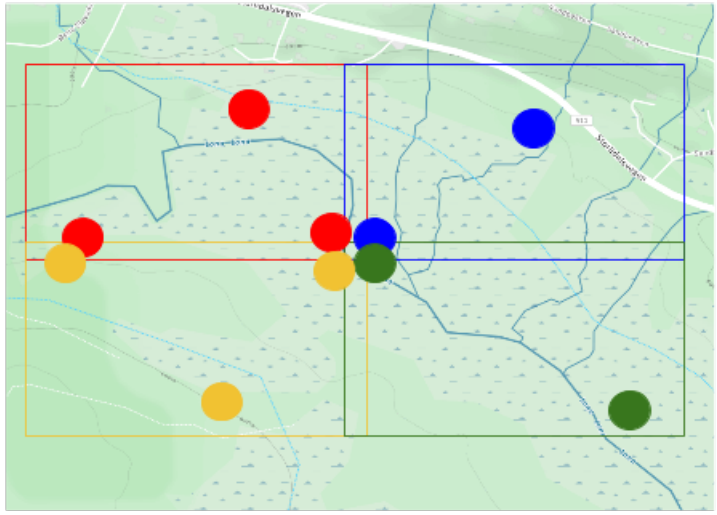

Figure 2.6: The Four Pictures and Their Findings

The picture above shows how four pictures overlap, and how some of the herds overlap. The red picture and the yellow picture have two herds that overlap. This is the same herd, but taken at different times. In the middle there are four findings, which is the same herd. All of the four pictures have managed to capture the herd.

2.1.3 Pecora/P

Pecora/P gives the presentation of the results gathered from the drone and the picture analysis. It is Pecora/P that interacts with the users and gives them the information they need. Pecora/P focuses on the best way to display the results so the users can understand them. Pecora/P will use a map and display the herds on the map given the coordinates given by the picture analysis. Information such as time, number of animals in the herd and species, must be easy for the user to find. Pecora/P is all about a user friendly presentation, and displaying the herds and their information in an unambiguous manner. Pecora/P will not only display the herds, it will show the paths and areas covered by the drone. That will give the users the ability to determine where to search next. A timeline will be included to let the users see older results from previous flights. Then the users can see where the sheep have been, as well as where they are now. The users can use this information to determine where the sheep are heading. Every other information needed to find the sheep in the wild afterwards will be presented.

If there are several findings of the same herd, due to overlap in the pictures (Figure 2.6), Pecora/P should combine them to represent one herd, and not four separate herds. The

result would look more like Figure 2.7.

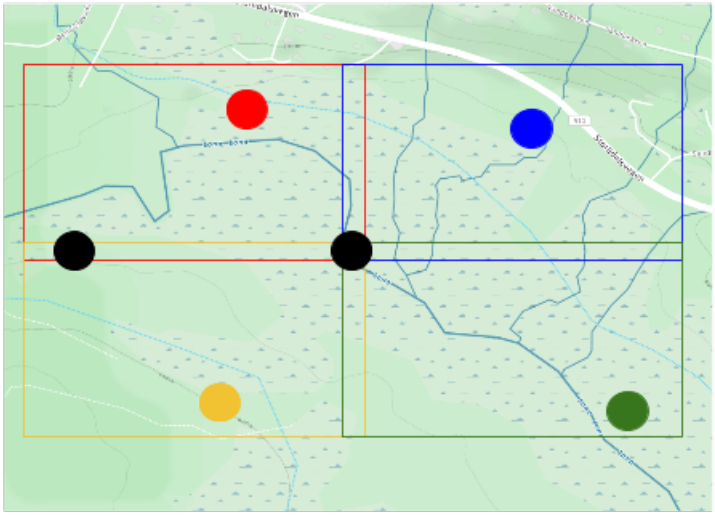

Figure 2.7: The Four Pictures and Combined Findings

The black spots in the figure above represent one herd that are combined by several findings from the picture analysis. The findings have been combined to show the user that it is in fact just one herd, not two/four separate herds. Pecora/P will display each herd, and the original picture they were captured. Pecora/P will handle different species by giving them different icons and specifying what types of animals each finding include.

Pecora/P will display the flights, and connect each finding to the flight they were produced. The map will therefore only show the flights and the findings and not the pictures. The pictures will be displayed after a finding have been checked. The idea is to give the users enough information without spamming them with useless information. The pictures will just take up space on the map, and distract the users. The goal is to let the user see the information about each finding and flight by clicking on them, and hide the information of the unchecked findings and flights. This gives the user the choice to pick the findings or flights they would like more information about, without being distracted by the other findings. Displaying a list of all the information about every single finding can make the user give up. It is just too much information at the same time.

After Pecora/P have presented the result and the information available by the drone and picture analysis, the user should have everything they need to walk into the wild and locate their sheep.

Chapter 3

Architecture

This chapter describes the architecture and the requirements of Pecora/P. The requirements can be found in Section 3.1, and then a short description of the stakeholders and their concerns can be found in Section 3.2. The architecture will be demonstrated by different architectural views, to be found under Section 3.3.

3.1 Architectural Significant Requirements (ASRs)

This section covers Pecora/P's requirements. The technical requirements will be covered first in Subsection 3.1.1, then the functional requirements will be covered in Subsection 3.1.2 and lastly the quality attributes will be covered in Subsection 3.1.3. The requirements listed in this section can be seen as guidelines to answer the main research questions found in Section 1.2.

3.1.1 The Technical Requirements

The information system will be built using HTML and JavaScript, with Leaflet, MapBox and Bootstrap.js as extended libraries, Flask and PyMongo as extended libraries for the back-end and MongoDB as the database-server. The developer has experience with using HTML, JavaScript and Bootstrap.js prior to this project, however, the rest of the technologies had to be learnt before the implementation could be started.

3.1.2 The Functional Requirements

The functions Pecora/P must provide to be a success are listed in Table 3.1. Each requirement in the Table has a codename, a description and a priority. The priority goes from low to high.

Table 3.1: Functional Requirements

Code	*Description*	*Priority*
FR01	Display the results from the picture analysis	High
FR02	Display the sheep in an unambiguous way	High
FR03	Display the flightpath in an unambiguous way	High
FR04	Display predators in an unambiguous way	Medium
FR05	Display different species in an unambiguous way	Low
FR06	Display dead sheep in an unambiguous way	Medium
FR07	Display a map containing all the findings	High
FR08	Use a map with contour lines, lakes, rivers, paths, roads and swamps	High
FR09	Use different colors to indicate different findings	Medium
FR10	Add a timeline of the findings	Medium
FR11	Display older findings in weaker colors	Low
FR12	Display all the information needed to locate the sheep	High
FR13	Display the time and coordinates of the findings	High
FR14	Display the species of the animals	Low
FR15	Display number of lambs in the herd	Low
FR16	Display the original picture taken by the drone	Low
FR17	Add a help guide	Medium

The goal is to present all available information the users need to find their sheep. The requirements are therefore on how to present the information and which information is necessary. A timeline-based presentation is also required as it would give the users information on where the sheep are and have been. That way they can also determine where the sheep are headed. A help guide is required to give new users the ability to understand what is presented. The map has its own requirements because the content of the map can help the users determine where to move to find the sheep. If the sheep are e.g. located on the other side of a lake, the user can use the paths to see where they should move to get to the sheep.

3.1.3 The Quality Attributes

The quality attributes are the quality requirements of Pecora/P that have to be delivered if the user are to be satisfied and keep using the program. The quality attributes are listed in Table 3.2. Each attribute has a description of why it is important for Pecora/P. The attributes are sorted on priority, from highest to lowest.

Table 3.2: Quality Attributes

Easy-to-use	Pecora/P must be easy to use, otherwise if the user struggles to use Pecora/P correctly or do not understand the presentation, Pecora/P will lose its purpose and therefore not be used.
Easy-to-learn	Pecora/P must have an easy learning curve, otherwise the user will find it time-consuming and will not see Pecora/P as useful.
Efficient	Pecora/P should work fast and efficient as sheep move around, the more instant the results are presented, the faster the users will be able to find the sheep in the wild. The ideal goal is to receive the results while the drone is still in the air.
Availability	Pecora/P must be available whenever the user need it. The data from old flights should always be available. By using the old data the users can determine where the sheep is headed, and they will be easier to find in the wild.
Mobility	Pecora/P should be able to be used at different locations and give the same result. The users have to be able to use Pecora/P wherever they are.

The quality attributes are there to make sure the user wants to use Pecora/P. If the user finds it hard to learn and use Pecora/P, then the user will give up, and seek other solutions. Without the user Pecora/P is useless. Pecora/P must be efficient because after the drone has flown and then landed, the presentation of the data must be available as fast as possible. Sheep move fast and they will not stand still for a long time. The longer it takes to present the data the longer the sheep will have moved and the information gathered during the flight will not be of much help. The ideal goal, though hardly possible, is to transmit the gathered information while the drone is flying. That way the system would be updated as fast as the drone found the sheep. The users are familiar with how sheep move around, and if data from earlier flights are displayed they can determine the most likely path of the sheep. If a sheep herd can be recognized as the same sheep herd from the previous day, then they can see where the herd is headed. If the user has an Internet connection when walking in the woods, though this may not always be the case, then they can use the information available to them while looking for their sheep.

3.2 Stakeholders and concerns

This section talks about the stakeholders of Pecora/P and their concerns. The stakeholders are a various groups of end users, developers, testers, supervisors and technical providers. The stakeholders and their concerns can be seen in Table 3.3

Table 3.3: Stakeholders

Stakeholders	Descriptions
End user	The end users are the users of Pecora/P. Their concerns are to understand how the findings are represented and to locate their sheep by using Pecora/P.
Developer	The sole developer is Julie J. Kirkhus. Her concern is to develop Pecora/P, making it represent all the necessary information, and making sure Pecora/P is functioning as planned.
Supervisor	The supervisor is Svein-Olaf Hvasshovd. His concern is to guide the developer and give constructive feedback on the development of Pecora/P.
Code tester	The code tester is Julie J. Kirkhus. Her concern is to test the code regularly to make sure all the features are functioning properly.
Usability testers	The usability testers are the sheep farmers Hallvard Storli and Steingrim Horvli. Their concern is to test the user interface and usability of Pecora/P and give feedback on possible difficulties.
Technical provider	The technical providers are the drone providers and the picture analysis developers. Their concerns are to add an picture analysis to Pecora/P with pictures taken by the drone. This is future development.

The table over tells which groups of people that can affect or be affected by Pecora/P. The end users are the farmers that need help with locating their sheep at the end of the grazing period. They will use the information provided by Pecora/P to locate the sheep, and hire the people that can fly the drone (the technical providers). The developer and code tester are the same person. That way the person developing Pecora/P can test the code regularly, and make sure Pecora/P are developed correctly. The supervisor will help the developer implement a correct and working program, and will oversee the development process of Pecora/P to make sure it is done within the time limit. The usability testers are two farmer who are invested in the Pecora system, and they will be the end users. They know what they need of information, and if the usability is understandable. They are therefore perfect for testing the user-friendliness of Pecora/P. The technical providers are the Department of Engineering Cybernetics who delivers the drones and can fly them and the future developers of the picture analysis. The technical providers will not deliver anything this spring, however Pecora/P are dependent on them.

3.3 Selection of Architectural Views (Viewpoint)

The architectural views of Pecora/P follows the guidelines from the "4+1" architectural view model (Kruchten, 1995). The "4 +1" architectural view model uses standard UML

diagrams (UML-diagrams, 2016) to represent the views. To represent the architecure of Pecora/P three of the architectural view models were chosen. The views are all presented from the different stakeholders perspectives. The views are the logical view, the process view and the physical view. Every notation of the views will be explained beforehand for a better understanding of what the views show.

3.3.1 The logical view

The logical view shows an overall structure of Pecora/P. The view will show the functionalities provided for the end users seen from an developers point of view. To represent the logical view, an UML Class Diagram is used (UML-diagrams, 2016). The logical view can be seen in Figure 3.1. The notation for the logical view is as follows:

- Class: Used to document the different classes. They are divided into three parts.
  - Name: The name of the class.
  - Fields: The fields of the class and their data type.
  - Methods: The methods belonging to the class and their input.
- Lines with blank arrow: Used to document classes that extends another class. The class without the arrow extends the one with the arrow.
- Lines with filled diamond: Used to document the relationship between classes. If the class with the filled diamond is removed the classes attached to it will also be removed.

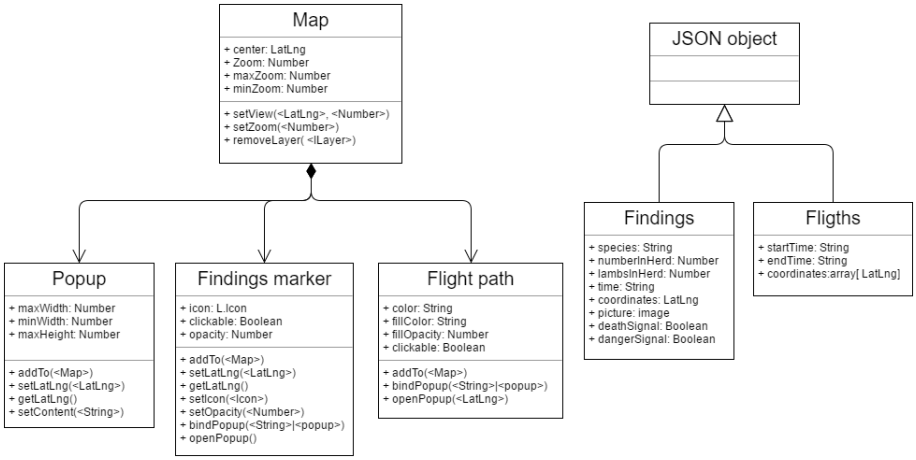

Figure 3.1: The Logical View

The class diagram is divided into two main groups, the map (to the left) and the JSON objects (to the right). The map is a Leaflet map object (Leaflet, 2016). The map object

have several predefined fields and methods which can be used. The center and Zoom are required when creating a new map object. The RemoveLayer() method can be used to delete layers attached to the map. This means the findings or flights can be deleted from the map by using the RemoveLayer(). Layers attached to the map can either be a Popup, a Findings marker or a Flight path. The Popup object is a small talk bobble filled with information that opens when someone clicks on the map. The information depends on the type of object that was clicked on. If a finding is clicked on the information about that finding are displayed in the Popup. The Findings marker object is a marker placed on the map at the coordinates given by the findings. Each finding have its own marker. The marker uses an icon or a picture to be displayed on the map. The markers can also change its opacity and bind a Popup object to it. The Popup object will be opened when the marker receives a click. The Flight path is a filled rectangle placed on the map based on the path taken by the drone. Each flight has its own path/rectangle. The path can bind a Popup to it, which will open when the path receives a click.

There are two extended JSON objects, the Findings and the Flights. The Findings objects contain every available information about the findings. The Findings marker can use the information in the Findings object, as well as the attached Popup object. The paragraph next to the map which is displaying the information about the findings can use the Findings object to obtain the information. The Flights object contain all the available information about the flights. The paragraph next to the map, the Flight paths and their attached Popup object can all gather the correct information about the flight from the Flights objects.

3.3.2 The process view

The process view shows the processes of Pecora/P. The process view is seen from the users point of view and shows every state the user can meet when using Pecora/P. To represent the process view, an UML Activity Diagram is used (UML-diagrams, 2016). The process view can be seen in Figure 3.2. The notation for the process view is as follows:

- Blank circle: Used to indicate the start node.
- Filled circle: Used to indicate the end node.
- Box: Used to show the different activities/states of the process.
- Diamond: Used to show the different choices the user can choose between.
- Arrow: Used to show the next state based on the previous activity or choice.

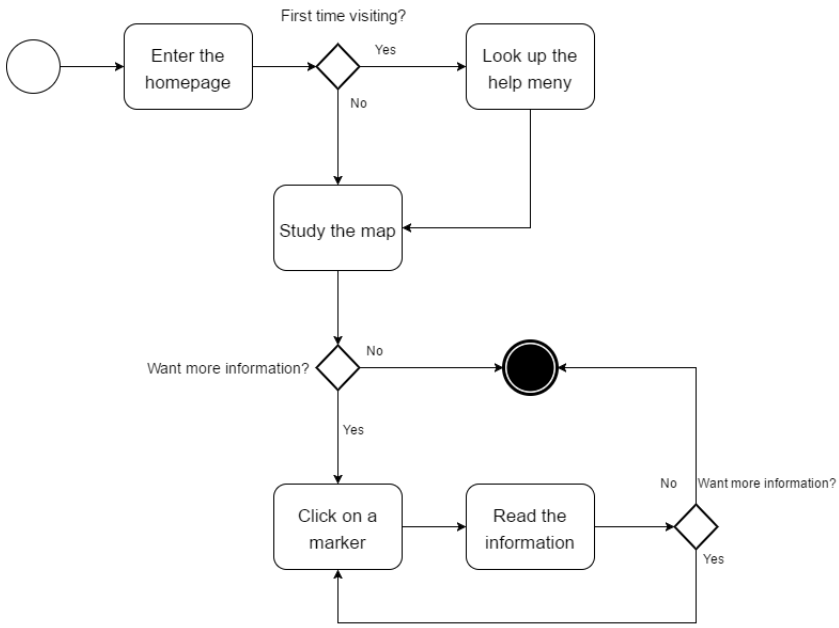

Figure 3.2: The Process View

The process view is seen from the users point of view and starts with the user entering the homepage. The map is displayed on the screen. If the user is new to the webpage, he may want to look at the help menu to learn the meaning of the figures on the map. Then the user will study the map themselves and view all the flights and their connected findings. If the user wishes to view more information about a flight or a finding, the user will have to click on the figure. The information about the finding/flight will then be displayed next to the map and the user can read the information. When the user have read the information he ma choose to click on another figure until satisfied or leave the webpage and thereby terminating the application.

3.3.3 The physical view

The physical view shows the psychical components and their connections in Pecora as a whole system. The view shows how Pecora/P is part of Pecora and how it is connected to the rest of Pecora. To represent the physical view, an UML Deployment Diagram is used (UML-diagrams, 2016). The physical view can be seen in Figure 3.3. The notation for the physical view is as follows:

- Box: Used to show the components. The components can be artifacts or a subprogram.
- Arrows: Used to show the connection between the components. The arrows follows where the data is sent.

- Cylinder: Used to show the database.
- Big box: Used to show what components Pecora/P handles.

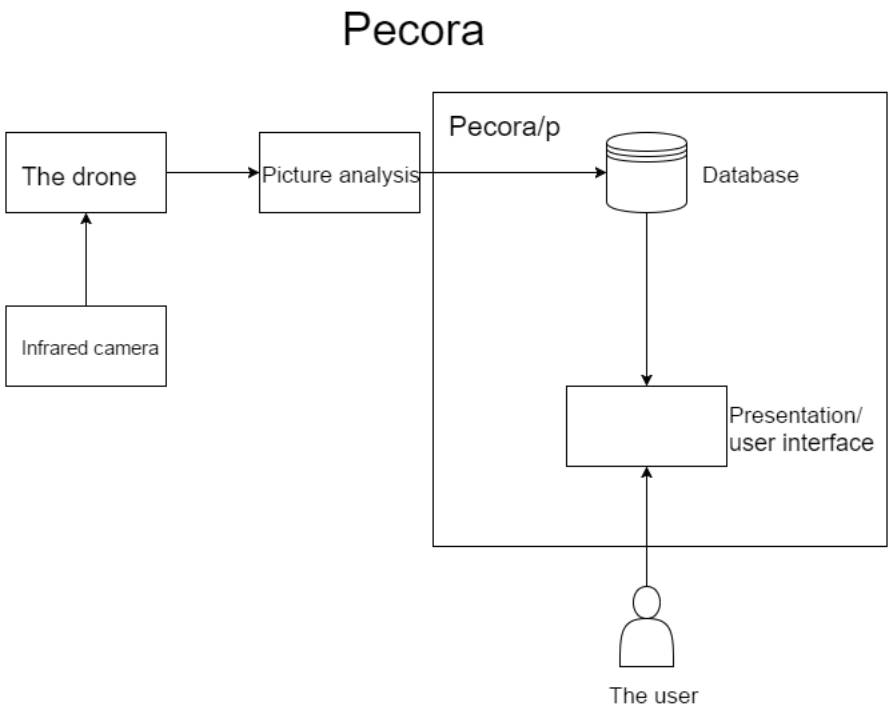

Figure 3.3: The Physical View

The components in Pecora are the drone, the infrared camera, the picture analysis, the database and the presentation (webpage). The drone carries an infrared camera that takes pictures during the flight. When the drone lands the pictures, time and coordinates are sent to the picture analysis. The result from the picture analysis will be entered into the database. The webpage will then get the data from the database, and display the information to the user.

Chapter 4

Implementation

This chapter presents Pecora/P's implementation and the technologies used to achieve its requirements. The goal is to find the best way to present necessary information needed to locate sheep in the wild.

A choice was made to implement a webpage for the representation. The main reason being that a webpage is accessible anywhere as long as there is an Internet connection. Internet are becoming more and more available all over Norway, and it was therefore not though as an obstacle, and a webpage were chosen as a solution. Another good reason is how easy it is to find documentation on webpage development. By developing a webpage application the users do not have to install a program. The user will need a browser however to run the webpage, but in these modern times, one can predict with great likelihood that the user already has a browser.

A webpage application can be split into three main parts: the database, the back-end and the front-end. Each part will have a discussion on the technologies applied. Why and how the technologies were implemented for Pecora/P will be covered. The first part, the database, will be covered in Section 4.1, then the back-end covered in Section 4.2, and finally, the front-end will be covered in Section 4.3.

The database, back-end and front-end are hosted by a DigitalOcean server running Ubuntu and using Apache2 as the Web server (DigitalOceans, 2016).

4.1 The Database

This section will talk about Pecora/P's database and the choices made when implementing of the database. Subsection 4.1.1 will present an Entity–relationship (ER) diagram of the database. The ER-diagram will show the type of data to be stored and how it is stored. Then the MongoDB database-server, which is the chosen database-server, will be presented in Subsection 4.1.2.

Before we start, it is important to understand what type of database is required. One of the central database requirements is to store pictures, since the pictures will be displayed in the front-end. Every finding found in the pictures should be able to point to the original infrared picture taken by the drone. The original picture should be displayed on the web-page so the users can study the pictures themselves. The database will also need to store the time and coordinates in a front-end readable way.

Another requirement is that the database must be modifiable. The Pecora system includes a picture analysis, which is not yet implemented. This means the database must be open for whatever changes that might appear after the picture analysis is completed. The format of the data received by the drone has not been fully determined per se. The database must be modifiable for changes that might occur related to the drone.

4.1.1 ER-diagram

The entity-relationship diagram will show the tables for Pecora/P with their attributes listed. The ER-diagram can be seen in Figure 4.1. The ER-diagram will not follow all the standard rules of a regular database, but will contain attributes a MongoDB database can store. This means a list and a picture can be an attribute. Under the ER-diagram comes a full description of each attribute. The description will include why this attribute is necessary and the attribute format.

The ER-diagram is composed of two tables: The Flight table and the Findings table. Each entry in the Flights table stores all data about one flight. The most important data is the start-time and end-time of the flight and a list of coordinates calculated during the flight. Each entry in the Findings table stores all data about one finding. The most important data is what species the animals detected are, how many animals are in the herd, the time when the picture was taken, the coordinates of the herd/animal and the original picture taken by the drone. The database should be simple and it should be easy to change the structure if needed. An example is to add a picture table that contains the picture, a picture id and the time when the picture was taken instead of storing them in the findings table. The picture database has not been added for Pecora/P this round, because of the delayed result of the picture analysis.

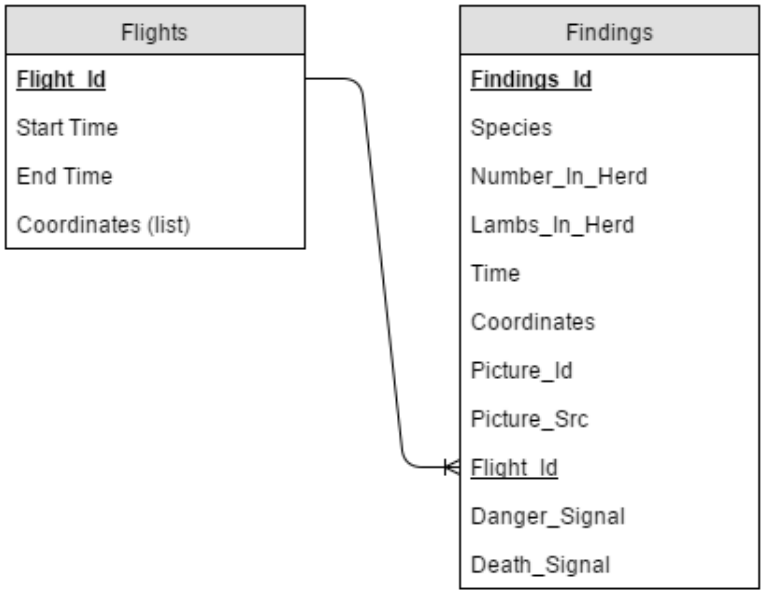

Figure 4.1: ER-diagram

The Flights table

The Flights table contains four attributes as follows:

- ***Flight_Id***: A unique id used as the primary key for one particular flight.
  format: unique integer
- ***Start_Time***: The time the flight started.
  format: string, "DD.MM.YYYY HH:mm"
- ***End_Time***: The time the flight ended.
  format: string, "DD.MM.YYYY HH:mm"
- ***Coordinates***: A list of all the coordinates from the flight.
  format: array of latitude and longitude pairs, [[lat,lng][lat,lng]]

The Findings table

The Findings table contains ten attributes as follows:

- ***Findings_Id***: A unique id used as the primary key for one particular finding.
  format: unique integer
- ***Species***: The species of the animals detected.
  format: string, e.g. "Sau"

- ***Number_In_Herd***: The total number of animals in the herd, regardless of species and age.
  format: integer
- ***Lambs_In_Herd***: The number of lambs in the herd.
  format: integer
- ***Time***: The time when the photo is taken.
  format: string, "DD.MM.YYYY HH:mm"
- ***Coordinates***: The coordinates of the approximate center of the herd.
  format: a latitude and longitude pair
- ***Picture_Id***: The picture's id.
  format: unique integer
- ***Picture_Src***: The binary representation of the picture.
  format: string, a base64 string
- ***Flight_Id***: The id of the specific flight during which the findings were produced.
  format: unique integer
- ***Danger_Signal***: Signals that a predator is detected.
  format: Boolean, true or false
- ***Death_Signal***: Signals that dead animals are detected.
  format: Boolean, true or false

The time attributes in both tables will be stored as strings. This is because the front-end will convert the string into a Moment.js object (a javascript time object) (Momentjs, 2016). Moment.js will be explained in Section 4.3. The string will be of the form "DD:MM:YYYY HH:mm". The DD represent the day of the month (1-31), the MM represent the month number (1-12), the YYYY represent the 4 digit year, the HH represent the hour (0-23) and the mm represent the minutes (0-60).

The flights will not send every single coordinate calculated during the flight. That would take up to much space and is not necessary for creating a good representation of the flight on the webpage. A few coordinates, minimum the start and end positions will be sufficient. If the flight follows the path described in Subsection 2.1.1, then the start and end positions of the flight should be enough for the representation as the area covered will be similar to a rectangle. Drawing a rectangle in the front-end only need two corners, that means the start position and the end position. If the end and start positions are on the same side of the rectangle, then the database must store several coordinates. The best coordinates to store are those in every turn in the flight path.

In the findings attributes there is one attribute informing of total number of animals in the herd (Number_In_Herd) and one to tell the number of lamb in the herd (Lambs_In_Herd). The reason is that in the fall, the lambs have grown to almost full adult size. It will be a

difficult task trying to determine the difference between an adult sheep and a lamb. The table therefore has to store the total number as the most important number and have the Lambs_In_Herd attribute in case the picture analysis manages to detect who are lambs. The total number includes the lambs and should be the sole indicator for how many sheep and lambs are in the herd.

The chosen database-server stores pictures as their binary representation. This means the picture must be converted into a binary string. For more information on the storing of pictures see Subsection 4.1.2. It is time consuming converting pictures into binary strings. It is therefore only one finding in the simulated data that has stored a picture. The picture is a small icon of a sheep and is only used to demonstrate how pictures can be stored and later used in the front-end. With just one picture in the database the Picture_Id is not included in the implementation, only the ER-diagram. A Picture_Id should be added after the picture analysis is implemented. A picture id will simplify the database querying if one wants to see all the findings attached to one picture.

In the current development of Pecora/P, the database will only allow one picture per finding. When Pecora is completely finished, there may be a need to expand the database to allow several pictures per findings. Adding several pictures to one finding can easily be done with MongoDB (see below). When the drone takes pictures there might be an overlap, and if a group of sheep are in the overlapping areas, there can be up to four pictures relating to one finding. All the pictures should then be added to the database, and one finding will have several pictures stored. This was however not implemented for Pecora/P as it was assumed unnecessary to add such a feature without knowing if it would be possible to determine if one sheep herd is the same as another from an earlier day. The goal however is that to make this possible and that every picture showing the same sheep herd will be included with the findings.

The foreign key Flight_Id in the Findings table is not implemented for Pecora/P. The Flight_Id attribute in the Flights table are auto generated by the database-server when inserting simulated data without an id-attribute. The Flight_Id was therefore not included in the Findings table as it would be time consuming finding the id for each flight and then to connect each finding to the right flight. The reason to have it in the ER-diagram is for when the real data is inserted into the database, then an id should be generated and added to both the Flights table and the Findings table.

4.1.2 MongoDB

MongoDB is an open-source, non-relational database that stores the data in collections (tables) and documents (records) (MongoDB, 2016). One collection can store several documents. Documents in the same collection are not required to have the same schema. The only attribute every single document must have is an id. The id must be unique and used as the primary key. If the document is missing an id attribute MongoDB will generate an id and add it to the document. Documents can vary in many ways within the collection, and they can store lists and even include other documents. MongoDB was chosen because of the easy way to change the document's schema. Pecora is not fully implemented, and

so the database structure must be easy to modify when every part of Pecora are to be connected. MongoDB can also store pictures and the documents can be converted into front-end readable objects.

The MongoDBs documents are stored as BSON objects (BSON, 2016). A BSON object is similar to an extended JSON object (JSON, 2016). Extended meaning that BSON objects allow storing data types the JSON objects do not allow. JSON objects store a subset of BSON objects. When querying the collection, the MongoDB will return a list of BSON objects. The list can simply be converted into JSON objects the front-end can use.

Simulated data

This project had a time limit and there was no time for collecting real data. Some simulated data was therefore created. The simulated data was stored as JSON objects to be inserted into MongoDB. For simplicity, the tables were split into two files containing their documents respectively. The JSON objects have the same structure as seen in the ER-diagrams. The documents in both the Flights table and the Findings table were stored without an id attribute since MongoDB will generate a random id for the documents.

When creating the flights documents it was not seen as a necessity to add more than the start position and the end position as coordinates. In the Findings Table only one document has stored a picture. The reason for this is that it was seen as a waste of time and effort to convert several pictures into binary strings. The picture included was converted into a base64 representation of a binary string. JSON objects to not allow binary string so a Base64 representation is needed. Because the pictures were ignored the Picture_Id was ignored. The tables was saved as flights.json and findings.json and sent to the server where the MongoDB is running.

Setting up MongoDB

MongoDB was installed on the server where the front-end, the back-end and the simulated data were stored. When installed the simulated data could be inserted into the MongoDB. MongoDB uses Mongoimport to insert JSON objects into the database. Mongoimport can be run from the command line. Mongoimport need to know the name of the database and which collection the documents should be inserted into. The JSON files must also be included.The following command line inserts all the JSON objects in Flights.json into the collection named Flights onto the database named Pecora.

```
$ mongoimport --db Pecora --collection Flights --drop --
    file Flights.json
```

When this command runs, a message of either success or error will occur. If a success message arrives the MongoDB shell can be used to view the database. If an error occur, it is most likely something wrong with the format in the JSON objects, check the JSON

objects and try again. The shell allows queries to interact with the collection. Now the data is stored and ready to be used.

Using Mongoimport is an easy way for inserting new data to the database, though it requires being on the server. For Pecora/P it is perfect, though for Pecora as a whole other approaches can be used. It all depends on how the picture analysis is implemented, and how the data from the drones are stored. MongoDB allows for several approaches and an example is to use Python to talk to the database. The next section will discuss back-end technology to gather data from the database, and the same technology can be used to insert data into the database.

4.2 The Back-end

This section will describe the back-end of Pecora/P. The back-end is the data access layer of Pecora/P and handles how to extract data from the database, manipulate the data and then send the data to the client. The client running the HTML and javascript, will not be able to connect to the database. Most servers do not allow for client-side interaction as this is a huge security risk. A REST interface can be used as a middleman between the server and the client.

4.2.1 REST interface

MongoDB recommends several REST interfaces that work perfect with MongoDB. The REST interfaces can be used by different programming languages, such as Python, java, Ruby and Node.js. Python is a simple language and easy to install and set up and therefore a good choice for a language (Python, 2016).

Eve REST interface

MongoDB recommends a Python REST interface named Eve (Eve, 2016). Eve is often used for its easy and fast set up. Eve must be set up on the server and it needs two files: settings.py and run.py. The settings file tells Eve what database to connect to. The host, port number and the name of the database must be put in the settings file. The collections in the database is also put in the settings file. Each collection can be set up with a schema that tells what fields must be included and each field's data type. The run file runs Eve and looks for a settings file to connect to the database. Eve then runs at a given host and the client can connect and send AJAX calls to the REST interface and receive data from the MongoDB. The run.py file only include a few lines of code as seen below:

```
from eve import Eve
app = Eve()
if __name__ == '__main__':
    app.run(host=<host>, port=<port>)
```

The <host> and <port> is the host and port numbers of the server running MongoDB

respectively. Eve is great for a standard REST interface with an ready to use framework. However the developer does not have much control over the interface. This is because of its many build-in features and settings, and the documentation is very limited on how to change them.

As mentioned, the client needs only send an AJAX call to receive the data from the server. This does not always work, as the browsers often will send a cross-domain error, which means the server does not allow clients on other domains to request AJAX calls. In the case of Pecora/P, the server did not accept the client's AJAX calls. While trying to fix the cross-domain error, the lack of documentation made it too time consuming to continue and an alternative to Eve was discussed.

Flask-PyMongo REST interface

An alternative to Eve is Flask REST interface (Flask-PyMongo, 2016). Flask does not handle AJAX call and can therefore fix the cross-domain error given by the browsers. Flask cannot talk to MongoDB by itself, and PyMongo is needed. PyMongo is a Python library that can interact with MongoDB. PyMongo can be used to query for documents as well as insert new data into the database. Flask-PyMongo builds the bridge between Flask and PyMongo, and Flask can now be used directly towards MongoDB.

Flask need an __init__.py file to run and start the REST interface. The code snippet under shows how Flask runs is initialized and PyMongo then runs Flask.

```
from flask import Flask
from flask.ext.pymongo import PyMongo
app = Flask(__name__)
app.config['MONGO_HOST'] = 'localhost'
app.config['MONGO_PORT'] = 27017
app.config['MONGO_DBNAME'] = 'pecora'
mongo = PyMongo(app)
if __name__ == '__main__':
    app.run(host='178.62.143.17', port=27017)
```

The Flask application uses app.config to determine what database Flask can connect to. To specify that it is a MongoDB database MONGO_ * is used as a prefix. As with Eve, the host, port-number and the databases name must be included. Running the __init__.py will now run the REST interface on the servers host and port-number. Flask then uses routing to determine what shall happen when the user enters a specific address. For Pecora/P there is only one address: The homepage. Flask gathers the data from MongoDB and then sends it alongside a Jinja2 template. Jinja2 is a template language for Python. The render template function takes the Jinja2 template containing empty variables and fills the variables with the gathered data. See the following code snippet for a better understanding.

```
@app.route('/')
def home_page():
    findings = mongo.db.findings.find()
    flights = mongo.db.flights.find()
    return render_template('index.html',
        flights=dumps(flights), findings=dumps(findings))
```

The code snippet show what happens when the client requires the address: `http://178.62.143.17:27017/`. Flask start by sending a query to both the findings collection and the flights collection. MongoDB returns a python.cursor which can be iterated over to get one document out of the collection. Then Flask returns the render_template function with a template called index.html and sends the gathered data as variables called findings and flights. The dumps function means that the gathered data are converted into JSON objects. In the index.html two lines as the code snippets show will be filled with the JSON objects respectively.

```
    var flightData = {{flights|safe}};
    var findingsData  = {{findings|safe}};
```

The client can now read the flightData and findingsData as regular JSON objects. JSON objects are easy to go over iteratively and the data can be used by the front-end to display the flights and findings on the webpage.

For making sure the REST interface run at all times, the server can use Upstart. Upstart is a program that runs on Ubuntu and it's tasks is to start other task while the server boot. To make sure the init file starts at boot a *.conf file needs to be added to the /etc/init directory on the server. The configuration file looks as the code snippet shows.

```
    description "run flask on startup"
    start on startup
    task
    exec python <path to init.py>/__init__.py
```

The <path to init.py> is where the init file is stored on the server. Then reboot the server, and the Flask REST interface will run without having to manually start it, and it will run as long as the server is up.

4.3 The Front-end

This section will describe the front-end of Pecora/P. The front-end is the presentation layer of Pecora/P and handles the interface between the user and the back-end. The front-end runs on the client and has the main task of displaying the findings from Pecora on an user-friendly way. A text-based information system is a good way of showing information,

though it can be a bit formal and boring. It was therefore decided to show the findings on a map. By using a map the findings can be placed on the map at correct coordinates, and the user will not need to use a different map to look up the coordinates.

There are several ways of displaying maps on a website. It can be a still photo of an area or it can be a scalable map, which can show the whole world or just a street. The latter is a better choice for Pecora/P. In that way Pecora can be used wherever the users are and is not tied to a specific place. Because of the time limit for the project, the first and working technologies were chosen. For implementation, the map Leaflet (Leaflet, 2016) and MapBox (MapBox, 2016) were tested. Both were easy to implement, and so due to the short time limit no further technologies were tested. The navigation bar, the information panel and the rest of the webpage was developed using bootstrap.js, HTML and Javascript.

4.3.1 Gathering the simulated data

Before initializing the map the data from Pecora must be preprocessed. By using MongoDB as the database-server the data is on the standard JSON format. The javascript can read JSON objects as a lists, and can easily collect the data from the objects. Most of the fields in both the Flights table and the Findings table are stored as strings or numbers and are very straight forward. The coordinates do not need manipulation, as Leaflet uses latitude and longitude to place the markers and figures on the map.

The time attributes requires manipulation since they are stored as regular strings and are preferred to be stored as Moment.js objects (Momentjs, 2016). Moments are dates that can be parsed, validated, manipulated, and displayed in javascript. Moment.js takes in a string containing the time and the format of the string to create a Moment object. An example can be Moment("12.12.2016 13:37", "DD.MM.YYYY HH:mm"). The Moment objects can be compared with each other and by using javascript the dates can be sorted from newest to oldest. This is crucial if a timeline of the findings is wanted. The moments can be displayed in a more fancy way than the original string. Instead of "12.12.2016 13:37" the information panel can say: "12. Dec 2016 13:37", where the latter is more user-friendly.

The picture is stored as a binary string, and javascript will convert the binary string to a png file and display it on the webpage.

4.3.2 Implementing the map

Pecora/P has a few requirements for optimal use of the map. The map must be a topographic map that includes contour lines, lakes, rivers, paths, roads and swamps. Mapbox delivers multiple maps and has a map-studio which lets the developers create their own maps. For this project there was no need for creating a new map as Mapbox already has a map meeting all the requirements. A map called Outdoors. Leaflet is an open-source javascript library for interactive maps. Leaflet gives the developers a full set of useful features to be used on the maps from Mapbox.

Leaflet lets the developers add pop-ups, markers, lines and figures onto the map. All can be placed by coordinates given in latitude and longitude. This means all the items will be placed on the map and not the screen. If the map moves to the right, the item follows. For Pecora/P the map must be able to place the findings and let the users know what type of animal is found. The finding can be of sheep, predators or other type of animals such as reindeer. Markers and circles were both tested. After a user-test markers was chosen. See more about the testing in Chapter 5 - Testing. Leaflet lets the developers customize their own icons on the markers. That way sheep, different animals and predators can have their own icon. Different icons let the user see what kind of animal is on the map without having to click on the markers.

Leaflet gives the user the ability to click on the map, markers and figures. Clicking an item will cause a function to run. Each group of items have their own function. What happens when a user clicks on an item is up to the developer. If a user clicks on a findings marker, more information about the findings can be displayed as a pop-up, or in an information panel next to the map.

For adding a map in Leaflet, a map object must first be attached to a div. The div is placed where the map will be showed on the webpage. A Leaflet map object needs two arguments, the center coordinates of the map and the zoom-level where 0 is max zoom out-level. When the Leaflet map object is working a real map must be added. Leaflet fetches a Mapbox map by using an url where the id of the map and the access token for the developers user on Mapbox is included. Now the webpage will have a well functioning map.

The next task is to place in the flights and the findings at the correct places. The drones flight plan, discussed under Chapter 2 - Background and Motivation, should cover a rectangular area. The flights was therefore chosen to be represented as rectangles on the map. Each flight will be linked to a Leaflets rectangular object. All the rectangular objects are stored in an array called "flights". Each object has two sets of coordinates (start position and end position), a line color, a fill color and a fill opacity. The colors and the object's opacity is based on the date when the data was saved. The newest data will have the strongest, brightest colors. The object's opacity will decrease for each day since the photo was taken. Every object is then added to the Leaflet map object and when clicked on, the information about the flight are shown in a pop-up and in the information panel next to the map.

The findings will be placed on top of the rectangles representing the area covered by the flights. At the first draft all findings were to be represented as circles with different colors and different levels of opacities. Leaflets circle objects are similar to the rectangular objects. Circles need two parameters to be placed on the map, the center coordinates and the radius. The object's color and opacity are decided by the date similar to the color and opacity of the flights. The circles will need different colors to let the user know the type of animal found. The use of circles was not as user-friendly as first imagined. Based on the

results from a user-test, it was decided that circles was not good enough as an indicator of different animals. Hence markers with customized icons were chosen for Pecora/P.

Each finding is therefore presented as a Leaflet marker object. Each marker stored in an array called "findings". Each marker uses a picture as its icon. The icon markers are decided based on what type of animal is found. Four different pictures are used for sheep, dead sheep, different animal than sheep and predators. That way the users can see right away what the findings represent instead of having to click on them each time. The markers opacity will decrease for each day since the photo was taken. Hence, the newest findings will also be the strongest and the brightest. Each element in "findings" are added to the map and when clicked on, information about the findings are displayed in a pop-up and in the information panel next to the map. The pop-up will only show the species of the animal and the time. More detailed information will be displayed next to the map alongside the original infrared picture. Changing the detailed information was done in javascript by using document.getElementById().innerHTML to change the HTML to show the correct data belonging to the clicked marker.

Pecora/P is now fully functional and ready to be tested and used as a part of Pecora. Pecora/P has been tested by real farmers and the results from the test can be seen in the next chapter.

Chapter 5

Testing

This chapter will cover the tests run on Pecora/P and the results compared to the requirements. During this project Pecora/P needed to be tested to see if it meets the requirements specified in Chapter 3 - Architecture. This chapter will present the testers' evaluation of whether the various requirements were met or not. For each requirement the test results will be presented followed by a small explanation on how it was modified to fulfill the requirement. During the implementation of Pecora/P the code was tested regularly. If any errors occurred or any of the functionalities did not act as suspected or wanted, the developer fixed them immediately. In that way the code and program was always running as wanted without errors.

Whenever a feature of the program was running as expected, it was further tested for usability and functionality. The usability was regularly tested by the developer, as was the code and functionalities of the program. A user-test run with two farmers was planned and executed when Pecora/P was fully implemented.

5.1 Test Results

This section presents the feedback and test result from the user-tests run during implementation and from the usability test run with the two farmers. The tests were based on the functional requirements described in Subsection 3.1.2 and the quality attributes described in Subsection 3.1.3. The first test results presented are from the testing of each seperate functional requirement, and can be seen in Subsection 5.1.1. The results from the testing of quality attributes are presented in Subsection 5.1.2. The chapter will end with a discussion about the test results (Subsection 5.1.3).

5.1.1 The Functional Requirements Test Results

The tables below present the test results for each functional requirement. The executor and the date of the test is noted, as well as if it was a success or not, and any comment

the testers had is included. After the test, Pecora/P was changed based on the results. Pecora/P had circles instead of icons for the findings. Findings of sheep were presented as blue circles, predators were presented as red circles, dead sheep were presented as yellow circles and other animals were presented as purple circles.

FR01 - Display the results from the picture analysis

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They liked the idea of Pecora and liked how the findings and information were displayed. They wanted some changes to some of the features, but they agreed that Pecora is something they could use.

FR02 - Display the sheep in an unambiguous way

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Minor success
Comment:	They found the sheep circles on the map easily. However, the color of the circles could be improved. Blue on a green and blue map can be hard to spot. They suggested purple or black.

FR03 - Display the flightpath in an unambiguous way

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Minor success
Comment:	They found the flight paths, although blue is not a very clear color on a green and blue map. They suggested purple or black.

FR04 - Display predators in an unambiguous way

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Failure
Comment:	The predator marker should be different than the sheep marker, and not just the color. A different shape would be more unambiguous. A triangle was suggested.

FR05 - Display different species in an unambiguous way

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Minor success
Comment:	They found the different species, however they suggested to have a different symbol or shape than the sheep markers.

FR06 - Display dead sheep in an unambiguous way

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Minor success
Comment:	They found the dead sheep, however they suggested having a skull or a different symbol rather than a yellow circle.

FR07 - Display a map containing all the findings

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They thought a map was a very good approach for displaying the findings.

FR08 - Use a map with contour lines, lakes, rivers, paths, roads and swamps

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They thought the map showed a great amount of detail, and contour lines, lakes etc. was all on the map.

FR09 - Use different colors to indicate different findings

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Failure
Comment:	The circles with different colors were not a good representation of the findings. Different shapes and symbols should be used instead.

FR10 - Add a timeline of the findings

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They really liked how they can see the results from previous days. That way they could see where the sheep have been and where they are heading.

FR11 - Display older findings in weaker colors

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Minor success
Comment:	They understood that weaker colors were from previous days. Still, they would prefer a more distinct difference in the opacity to make a better discrimination between previous and newer findings, in previous days with the newest findings being even brighter and stronger.

FR12 - Display all the information needed to locate the sheep

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They could not think of any more information that would be necessary to locate the sheep.

FR13 - Display the time and coordinates of the findings

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They found the time and coordinates in the pop-up as well as the information panel next to the map.

FR14 - Display the species of the animals

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They could easily determine the species of the findings. They thought however, that a reindeer herd could be mistaken for a sheep herd, as they look very much the same seen from above with an infrared camera.

FR15 - Display number of lambs in the herd

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They could easily determine the number of lambs, although they thought it might be too hard to see the difference between a lamb and an adult sheep.

FR16 - Display the original picture

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They thought it would be great to see the picture themselves to determine if the picture analysis could be mistaken. During the test there were no picture shown, this was only explained where the pictures would be placed.

FR17 - Add a help guide

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Evaluation:	Success
Comment:	They wanted a help guide to understand the figures on the map. However, the help guide should not always be present. A drop-down menu would fit the need perfectly.

5.1.2 The Quality Attributes Test Results

The tables below presents the test results for each quality attribute. The executor and the date of the test is noted. The evaluation of the test and the time used on the test are noted as well. Any comment the testers had are also included.

Easy-to-use

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Time used	20 min
Evaluation:	Success
Comment:	They found it easy to use, even though the findings markers were not too clear on what the color meant before someone explained it to them. They managed to understand that they could click on the markers to get more information.

Easy-to-learn

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Time used	10 min
Evaluation:	Success
Comment:	It did not take long before they understood the colors of the circles, what the weaker colors meant, and that they could click on the findings and flights to get more information. There were, however, a need to explain this beforehand, and a help guide with explanations and a clearer description of the findings markers should be implemented to help the learning process for newcomers.

Efficient

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Time used	
Evaluation:	
Comment:	The efficiency can only be tested when the rest of Pecora is developed.

Availability

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Time used	1 min
Evaluation:	Success
Comment:	Pecora/P started right after they entered the correct webpage. Availability must be tested once more when Pecora is developed.

Mobility

Executor	Hallvard Storli and Steingrim Horvli
Date	02. May 2016
Time used	1 min
Evaluation:	Success
Comment:	Pecora/P worked at the Storli farm, which is one of the many farms where it will be used when Pecora is developed. Mobility must be tested once more when Pecora is developed.

5.1.3 Discussion of the Test Results

Before the test, some information was given to the testers. Predators, dead animals and other animals were included in the test to show how it would look like given that the picture analysis could manage to identify such events.

After the testers had tested Pecora/P, a few discussions came up. The biggest challenge was how the webpage presented the symbols representing the findings. All the markers were circles in different colors, as explained above. The testers thought it was not obvious that the red circles represented a predator. They would rather want a triangle or something that explicitly shows there are some dangerous animals lurking around the herds.

They were not totally satisfied with the blue circles representing a sheep herd either. They suggested a different color than blue, since blue did not give a good enough contrast to the green and blue map. Colors such as black or purple should be tested. Neither the dead animal markers were regraded as a good indicator that there were dead animals detected. The yellow circle should be swooped with a skull or an alert symbol. Something that will make sure the user understand that there has been a fatality. After a small discussion about what type of markers would be best, the decision ended on having purple circles with a white sheep as the marker for sheep herds. Yellow circles with a dead animal on (most likely a sheep) would be used to represent the dead animals. The predator marker was decided to be red triangles including a white exclamation mark. Other animals will have a black circle with a white paw. The webpage after these changes were implemented can be seen in Chapter 6 - Results.

The testers also mentioned that they found it difficult to see which findings were the newest and which ones were the oldest. They saw it, but it could be clearer which ones are the newest. The markers and the flights should both have a more distinct difference in their opacity based on the date they were produced. The testers were, however, very excited about the option of having the older findings included on the map. That way they could see where the sheep herds had been previous days. They could also determine where the sheep herds were heading. Hence, it would be easier to locate the sheep in the wild afterwards. There are no features so far that can determine if one sheep herd found today is the same sheep herd found the previous days. The testers, who are farmers, did however not see any trouble with this. They could look for herds with the same number of sheep in them, and based on the positions the days before, they could conclude that they most likely were the same herd. There were then a small discussion about the use of radio signals on the sheep. That would make it easier for the farmers to detect the same sheep herd during a period of several days. If radio signals are used, this information must be added to the information panel next to the map. To read more about Pecora and radio-signals see Chapter 7 - Discussion and Conclusion.

The testers thought that detecting dead animals would be nearly impossible. After a sheep or another animal dies, there will come large quantities of birds at the scene and they will eat the reminders of the dead animal. The drone must be flying over the dead animal within half an hour after it dies in order to see the animal. If the birds are captured on the picture,

the picture analysis can determine whether there is a dead animal there. However, this is highly unlikely, and after the birds are gone, there will only be scattered bones left. An infrared camera will not be able to detect the bones. Therefore, finding dead animals will be a hard job, but Pecora/P will still have the opportunity to place dead animals on the map, in case the picture analysis manages to find a way to detect them.

The testers did not see the need for identifying other animals, as they thought it would be both difficult and often wrong. This includes predators as well, although in some cases their size and the fact that they walk alone can be an indicator. They mentioned that seen from above with an infrared camera sheep and reindeer would look very much the same. They did not see this as a huge problem. If one out of many sheep findings are wrong, the end result would still be a huge help in locating the sheep. If there was a huge need to see the difference between a sheep herd and a reindeer herd, a regular camera could be used. A regular camera can determine if it is a sheep or a reindeer by the colors of the animals. If Pecora does not add a regular camera, then maybe the body temperatures of the animals could be of help, though it is highly unlikely, as most animals have approximately the same body temperature. It was, however, concluded that identifying other species are a big plus, but not necessary to give the farmers a good result. Pecora/P will always have the opportunity to display other animals and predators in case the picture analysis manages to identify them.

The testers were a bit skeptical about having the number of lambs displayed on the web-page. They thought it would be almost no chance of knowing if a sheep is a lamb or an adult. It was suggested that if the adult sheep wore a radio, that could signal the drone, then the farmers would know how many adult sheep were in the herd, and then the rest must be lambs. Again, this requires radios installed on every adult sheep. Pecora/P will always have the opportunity to tell the number of lambs in a herd in case the picture analysis manages to tell who are lambs.

The testers thought a help guide with legends for the markers will be a great idea. They did however think that there is no need for having a help guide displayed all the time. The help guide should be hidden until the users wants to use it. A drop-down menu in the navigation bar would be perfect.

After all, Pecora/P was a success. The testers got a better clue on what Pecora could bring and help in their daily life. They were very satisfied with what information was available. They could not come up with anything else to add of information about the findings nor the flights. Except for the need to change the markers to become less ambiguous, it was little the testers would change in Pecora/P.

Chapter 6

Results

This chapter will cover the results from the project. The result is a fully functional part of Pecora. Pecora helps the farmers locate their sheep during the grazing period, especially at the end of the period, when the sheep must return home. This project delivers the presentation part of Pecora. The presentation part is the interface between the users and the results produced by the drone and following picture analysis. The presentation should deliver the results from Pecora in an user-friendly manner. Unfortunately, the picture analysis was not implemented during the project period as a result of not finding another student, who could implement the picture analysis.

The presentation part of Pecora is called Pecora/P. In this chapter Pecora/P will be presented with different screen shots of the webpage. Some functionalities are added, e.g. picture analysis that manages to identify the species of the animals, marking them with a different marker than the sheep marker. If it is not possible to identify the species, then all the findings will be considered as sheep herds. This includes dead animals and predators as well. The number of lambs in the herd is also included. However, it might be difficult for the picture analysis to determine if it is a lamb on the picture or an adult sheep. The lambs tend to grow close to an adult sheep size during the grazing period, and it is therefore very difficult to see who are lamb and who are not.

6.1 Pecora/P

Pecora/P is a webpage for displaying where sheep and other animals have been seen based on pictures taken by a drone. The findings are displayed on a map along with the areas covered by the drone. Every group of animals is placed on the map, and when clicked on, information comes up in a pop-up and in an panel next to the map. The webpage allows the user to remove the flights from the map if they wishes. The webpage also includes a help guide in form of a drop down to help newcomers understand the markings on the map. Pecora/P will show all the findings from the past 8 days including the areas covered by the drone. All the flights are presented as purple rectangles. If the flights are new they

have a high opacity, if they are old they will have a lower opacity. The purple markers with a white sheep represent a sheep herd. The yellow markers with a sheep represent dead animals, most likely sheep. The red triangle means danger, and represent a predator. The black marker with the white paw represent other animals than sheep or predators, e.g. reindeers. The opacity of the markers decreases each day that passes. That means the markers with the brightest and strongest colors are the newest.

6.1.1 Pecora/P with Flight Paths

The home page of Pecora/P can be seen in Figure 6.1. The user will quickly see the map containing all the findings. The map is zoomed out so they are all viewable from the start. The areas covered by the drone are thereby default, but the user can choose to hide them by clicking "Uten flyrute" in the navigation bar. The user can zoom in on the map to get a closer look, and move the map around to look for more findings.

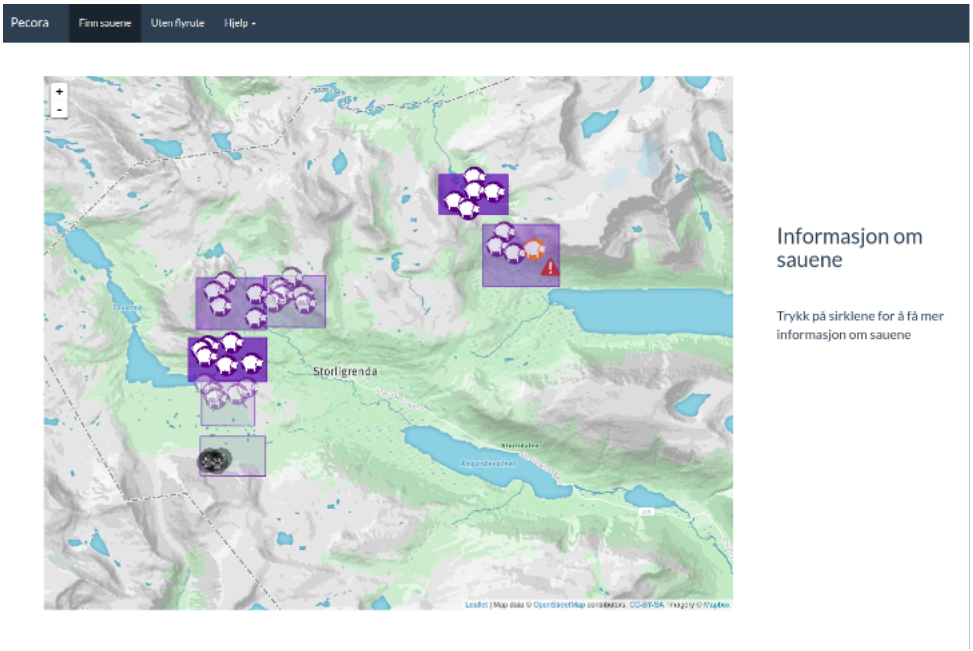

Figure 6.1: Pecora: The Home Page

6.1.2 Pecora/P without Flight Paths

The webpage without showing the flight paths can be seen in Figure 6.2. The picture shows how the map looks when the user have chosen to hide the flights, whereas everything else is still the same. The user can get them back by clicking either "Finn sauene", "Pecora" or refreshing the webpage.

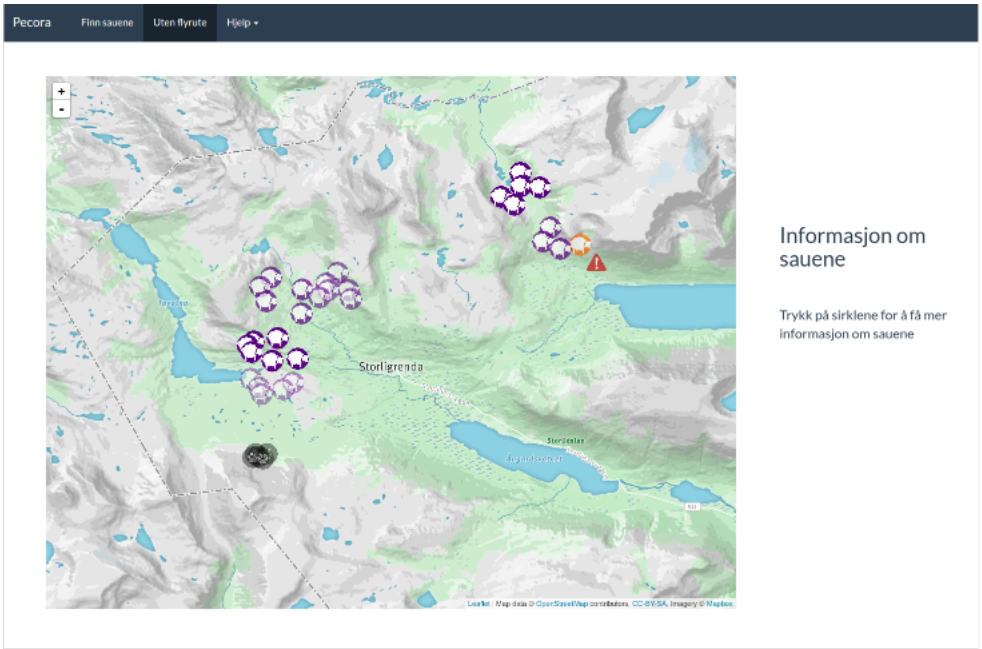

Figure 6.2: Pecora without Flight Paths

6.1.3 Pecora/P After Clicking on a Sheep Marker

Pecora/P allows the user to click on the markers as seen in Figure 6.3. The picture shows how a small pop-up "pops up" over the marker. The pop-up gives the user some information about the finding or flight. The pop-up can be seen more closely in Figure 6.4. The pop-up lets the user know the species of the animals, the total number in the herd, and if it is a sheep herd, the number of lambs will be given as well. If it is a predator or dead animals detected, the pop-up will tell the type of species and give a comment about the situation. The flight pop-ups give the start time and start position of the flight as well as the duration in hours. Next to the map is an information panel where the rest of the information can be found. The information panel can be studied more closely in Figure 6.5. The information panel gives the same information as the pop-up, as well as the time the picture was taken, the coordinates of the findings, and a comment about the findings. It will also include the original picture taken by the drone. Because of lack of real data, the sheep marker icons have been added to show how it could be shown on the webpage. The information about the flights include the information on the pop-up as well as the end coordinates.

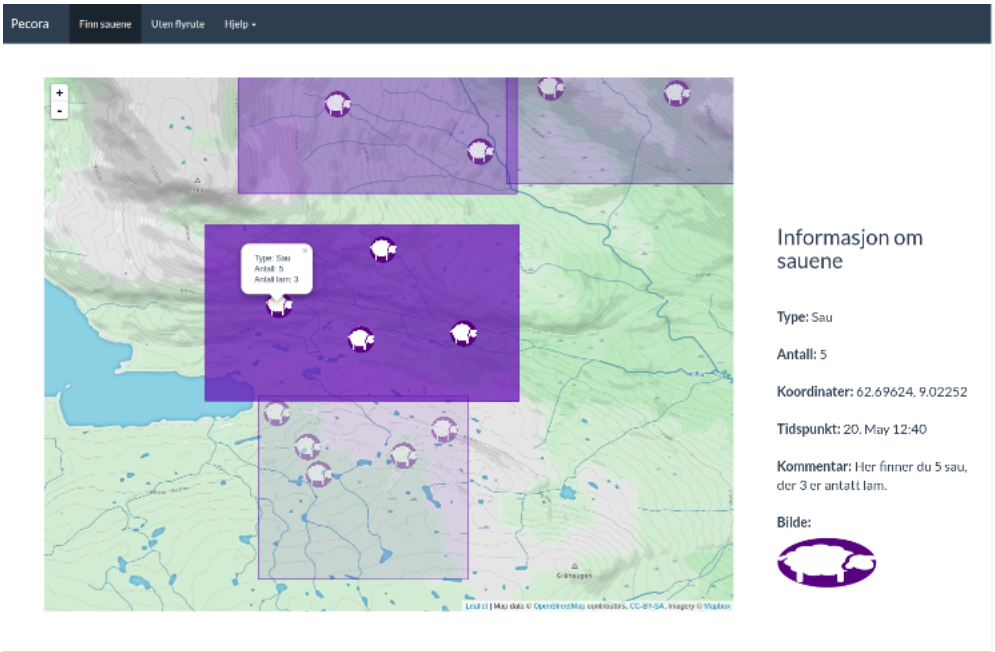

Figure 6.3: Clicked on a Finding with Picture

6.1.4 The Pop-Ups

A closer look at the pop-ups for flights (right) and two type of findings; sheep (left) and predator (middle).

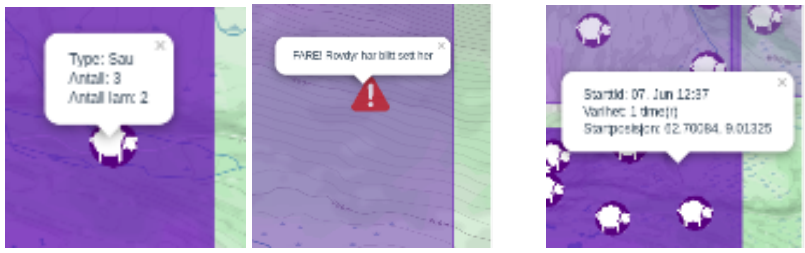

Figure 6.4: The pop-ups

6.1.5 The Information Panel

A closer look at the information panel for sheep (left) and flights (right).

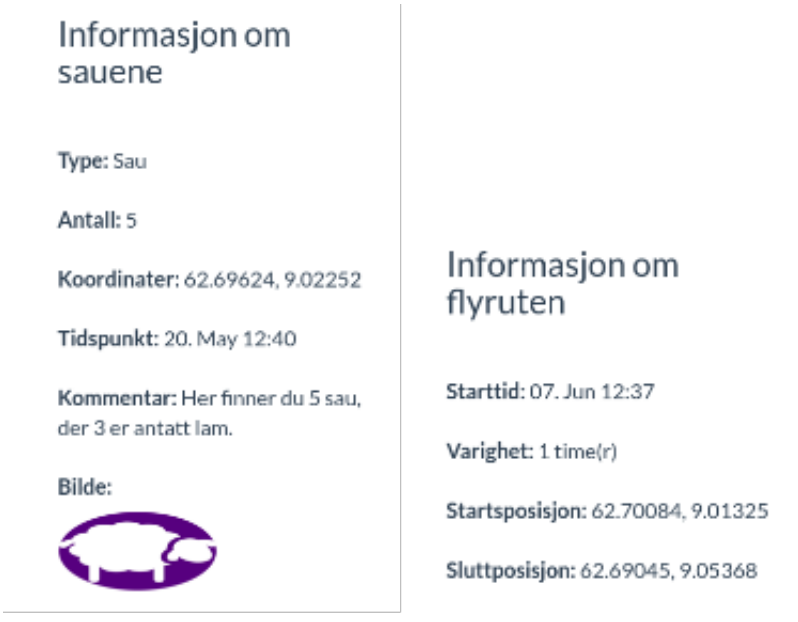

Figure 6.5: The Information Panel

6.1.6 The Help Guide

The help guide can be seen in Figure 6.6. The help guide explains the four markers with a little text next to the correct marker. It is simple and straightforwardly, and gives the user enough help to understand and use Pecora/P.

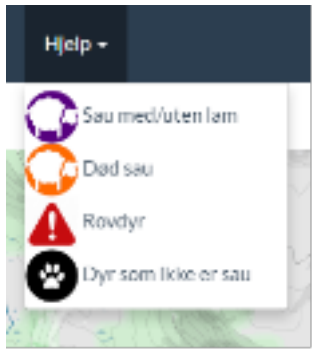

Figure 6.6: The Help Guide

Chapter 7

Discussion and Conclusion

This chapter will conclude this project and discuss the results of Pecora/P. The research questions will be answered in Section 7.1, and a discussion and recommendations for future work will be covered in Section 7.2.

The present project has delivered a user-friendly webpage called Pecora/P for helping farmers find their missing sheep at the end of the grazing period. The webpage is part of a system called Pecora. Pecora uses a drone that takes pictures of the grazing area. After landing, the drone runs a picture analysis and the results are presented on a webpage. By using a webpage, Pecora/P gives the user an easy-access application with no need for installations beforehand. Pecora/P gives the farmers information about where to find the sheep and related animals, and displays them on a map.

7.1 Conclusion

This project has examined different approaches for displaying animals findings for the system Pecora. The results have shown that a map filled with findings illustrated as markers in different colors has been a successful way for displaying the findings of sheep. The results have shown which information about the findings are necessary for the farmers to be able to locate their sheep in the wild. Based on the results obtained in the project, the research questions given in Section 1.2 have been answered.

What is the best approach for displaying findings of sheep and related animals efficiently to farmers?
Farmers and other potential users are more likely to understand a visual display of the findings than a textual presentation. Modern users are used to visual interactions daily, and a visual display is intuitive and fun to use. A textual display can be too long and complicated texts, and the information and can easily become boring. The information about a specific finding should only be displayed once the finding are checked or clicked. Giving the user all the information about every single finding of sheep and related animals, will put the

effort of sorting the findings upon the user, which can be quite bothersome in the long run. A map that shows the contour lines, lakes, swamps and other vegetation, can be helpful in case the sheep are heading towards areas where sheep are known to die, e.g. swamps are dangerous as sheep tend to get stuck.

What information is important to give the farmers about the findings of sheep and related animals?
The information about the sheep or related animals must be direct and specific. Too much unnecessary information will only confuse the users. The most important information is the time the picture was taken and the coordinates of the findings. With the time and the coordinates, the user have every information they need to find the sheep afterwards. Sheep tend to move around fast. The time is therefore crucial. The sheep will move further away from the given coordinates as time goes by. How far the sheep have moved can be calculated from the time when the picture was taken. The species of the animals and the total numbers are necessary information to give the users. If the animals found are not sheep, then the farmers can look away from that finding. If the animals are predators, then the farmers will be alerted and aware of the dangers in that area. The total number of animals is importent information as the farmers can then count the number of sheep at various time-points, and calculate whether there are some animals missing. The number of lambs can be useful in the beginning of the grazing period to see if the number of lambs corresponds to the number of lambs sent out to graze. In the later months of the grazing period, the adult sheep and the lambs will look too much alike in size, and the picture analysis will not be able to discriminate between lambs and adult sheep.

How can a map help display the findings of sheep and related animals in an unambiguous manner?
A visual display is better than a textual display. The findings can be match to specific locations, which can be placed on a map. The map will then give an overview of every findings of sheep and related animals in the area. Farmers are used to gather their sheep at the end of the grazing period with only a map and a pen. They will have to use a map when they walk in the wild to find the sheep. Letting the farmers see all the findings on a map, will save them some time as they will not need to convert the textual information onto another map afterwards. The map can have other fun ways of showing the findings than a textual application. Markers and figures can be placed on the map with different colors and sizes. If the farmers are only interested in sheep, they can look away from the markers with different colors than that of the sheep. This gives the user the ability to ignore certain findings of related animals and focus more on the findings of sheep. A danger icon can be useful to let the farmers know if a predator is observed near their sheep herd.

Overall, a map gives a great and straightforward visual presentation of the findings of sheep and related animals with the use of different markers to indicate different situations and findings. Time, coordinates, number of animals and their species are all necessary and important information that will help the users locate their sheep.

7.2 Future Work and Perspectives

This section will focus on recommendations for future work. It will discuss whether Pecora can be used in several fields and be expanded by using different technologies. Using technology to solve regular tasks is becoming more and more common. Pecora will be a huge help for farmers when they try to find their sheep, with the help of drones and information technology. Drones are a new and up-coming technology that more people tend to use. Pecora uses a thermal camera to locate the sheep.

Running a search algorithm on the pictures will not only find sheep. The result will most likely be any animal that are lurking in the area. Pecora is therefore not only usable to locate sheep. Reindeers and predators such as wolverine, lynx or wolf are just a few examples of animals the Pecora system may find. Pecora can be used not only in September and October, but it can be used the whole year around. This gives it the possibility to track animals all year around, though then a infrared camera alone will not be substantial. The warm weather during the summer will heat up the surroundings to a temperature similar to the body temperature of the animals. A normal camera must therefor also be used during the summer. When tracking animals it is important to understand their feeding habits, breeding and so on. Tracking the animals all year around will allow us to keep track of the quantity of different species. Every year thousands of sheep die due to predator attacks (nø; Bondevennen.no, 2012; **?**). If the predators that are harmful to sheep are tracked, then it would be easier to avoid future attacks. The farmers would see if their sheep are heading towards areas where predators have been observed.

To help with tracking, radio-signals sent from the animals may be used. In Norway, the use of radios attached to animals is a regular method for tracking. 75 lynxes, e.g. were tracked by radios in Norway by 2014 (NINA, 2014). Pecora can be expanded to use radios attached to the sheep. When the drone is flying over an area the radios will send a signal to the drone. The radio signal can send an unique id of the individual sheep. The radio signal will also send information about the number of lambs the given sheep has. Combined with a picture analysis, Pecora can also determine if there are any lambs missing. Every year several thousand sheep goes missing, most of them lambs. Tracking the number can help the farmers look for the missing sheep. Whether the animals are dead or not, the farmers will have an idea of the area where they went missing. If it is in an area known to have predators, it will increase their chance of getting reimbursement for the missing animals. Pecora could be expanded to not only detect radios-signals sent from sheep, but also radio signals sent from predators and other animals.

Using radio-signals means Pecora/P must be updated to show the ids of the sheep in a herd. This will make it easier for the farmers to track the movement of each herd. It may be represented by a dot on the icon, or something similar to show that the sheep have radios. The timeline can be expanded to show the movement of sheep with the same ids by connecting the markers with a line. The farmers showed a great enthusiasm for the timeline, and by linking the ids and timelines, the presentation will be even easier to use and understand.

Since radios attached to every adult sheep can be expensive, it is recommended to use the radios alongside the picture analysis. In that way the radios can be of big help, but not the only source of information. The sheep without radios can still be found, and so can lambs who have got lost and walked away from their mothers.

Pecora uses drones, which can be expensive for some users. An alternative is to use Pecora/P to display findings others have found when walking in the wild. Pecora/P can be extended to let users manually enter findings. The users can walk in the wild and come across a herd of sheep or a predator. Then they open Pecora/P and enter what they have seen, a marker will be displayed on the map so others can also see it. Pecora/P can be extended to be both a webpage and a mobile application, where users can enter the findings on the mobile without using Internet. When they are back in reach of the Internet, they can enter their findings into the webpage, probably by pushing a send button on the mobile application.

In summary, Pecora is an open and modifiable system with room for many future extensions and changes..

Bibliography

Anne Sigrid Haugset, G. N., 2010. Erfaringer med bruk av elektronisk overvakningsutstyr pa sau i 2010.
URL http://m.bondelaget.no/getfile.php/13117498/Bilder%20fylker/Nord%20-%20Tr%C3%B8ndelag/Dokumenter/Radibojeller%20notat%202010.pdf

Bondevennen.no, 2012. Sau på utmarksbeite, tap og tiltak.
URL http://www.bondevennen.no/sau-pa-utmarksbeite-tap-og-tiltak/

BSON, 2016. Bson binary json. Accessed: 2016-03-22.
URL http://bsonspec.org/

DigitalOceans, 2016. Digitaloceans. Accessed: 2016-03-22.
URL https://www.digitalocean.com/

DSR, 2016. Design science research.
URL https://en.wikipedia.org/wiki/Design_science_research

Eve, 2016. Eve python rest api framework. Accessed: 2016-05-25.
URL http://http://python-eve.org/

Flask-PyMongo, 2016. Flask-pymongo. Accessed: 2016-05-25.
URL https://flask-pymongo.readthedocs.io/en/latest/

Jim Hultgreen, R. I. S., 2002. Radio system for reliable data transfer.
URL https://brage.bibsys.no/xmlui/bitstream/handle/11250/137312/master_ikt_2002_hultgreen.pdf?sequence=1&isAllowed=y

JSON, 2016. Introducing json. Accessed: 2016-03-22.
URL http://www.json.org/

Kruchten, P., 1995. Architectural blueprints - the "4+1" view model of software architecture. IEEE Software 12 (6), 42–50.

Larsen, C. J., 2011. Kontrollerer ikke tap av sauer.
URL http://www.aftenposten.no/norge/Kontrollerer-ikke-tap-av-sauer-206940b.html

Leaflet, 2016. Leaflet. Accessed: 2016-05-25.
URL http://leafletjs.com

Lie, L.-E., 2012. Rovdyr og sau i norge og sverige.
URL http://www.bfrnorge.com/ROVDYR_OG_HUSDYR_I_NORGE_OG_SVERIGE.pdf

MapBox, 2016. Mapbox. Accessed: 2016-05-25.
URL https://www.mapbox.com/

Momentjs, 2016. Moment.js. Accessed: 2016-05-25.
URL http://momentjs.com/

MongoDB, 2016. Mongodb for giant ideas. Accessed: 2016-05-25.
URL https://www.mongodb.com/

NINA, 2014. Gaupa får skylda for urealistisk mange sauedrap.
URL http://www.nina.no/Aktuelt/Nyhetsartikkel/ArticleId/2431

NINA, 2015. Norge har god kunnskap om store rovdyr.
URL http://www.nina.no/Aktuelt/Nyhetsartikkel/ArticleId/3948

Python, 2016. Python. Accessed: 2016-05-25.
URL https://www.python.org/

UML-diagrams, 2016. The unified modeling language. Accessed: 2016-05-29.
URL http://www.uml-diagrams.org/