Geographical Based Wiki

Johannes Berggren

Bachelor of Science in Engineering Technology
Computer Engineering

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Geographical based wiki

Johannes Berggren

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

April 09, 2011
In this thesis the necessities needed in order to build a geographical based wiki has been evaluated. Users will get information connected to their location based on what others have previously contributed with. In order to evaluate the concepts explained in this thesis, a proof of concept application is built which will warn you of speed-cameras. The application has a few key features which could come in handy in further development of similar products and these features will therefore be evaluated in more detail. Using speed-cameras for the proof of concept prototype has been beneficial since they will not move. They can be compared against Trafikverkets (The Swedish road administration) positions and they are quite common. Using a server-script language to fetch the information from the database on the server side will allow the client application to be built for different platforms with the same backend. Since it is a crowd based application, it needs a way to filter less accurate registrations. Therefore an algorithm based on convex-hull peeling is used to calculate the position of the cameras. The work done in this thesis shows that group awareness can be used in applications to find information where sensors or other information gathering devices are too expensive or not cost efficient enough to use. Using group awareness leads to its own difficulties in itself by avoiding faulty information from users that do not know how to use the application or try to break it. This thesis explains the concepts and technologies which are necessary in order to create this type of application.
Working on this thesis is the final part of my education at Luleå University of Technology. I would especially like to thank my advisor Staffan Johansson at Neava for all his support. I would also like to thank Josef Hallberg my advisor at Luleå University of Technology, and the truck drivers at DHL Åkeri who helped me evaluating the proof of concept prototype.
## CONTENTS

### CHAPTER 1 – INTRODUCTION

1.1 Background .................................................. 1  
1.2 Goal .............................................................. 2  
1.3 Delimitations .................................................. 2  
  1.3.1 Requirements .............................................. 2  
1.4 Key Concepts .................................................. 3  
  1.4.1 Cloud Computing ......................................... 3  
  1.4.2 Crowd Computing ........................................ 3  
  1.4.3 Geographical Positioning Wiki ......................... 3  
1.5 Exploitation .................................................... 4  
  1.5.1 Travel Guide .............................................. 4  
  1.5.2 Road Assistant .......................................... 5  
1.6 Related Work .................................................. 5  
  1.6.1 GeoPedia .................................................. 5  
  1.6.2 LBWiki ..................................................... 5  
  1.6.3 Blackberry and iPhone CoDriver Application ........ 6  
  1.6.4 Road Sign Detection and Recognition ................. 6  

### CHAPTER 2 – SYSTEM DESIGN

2.1 Important Features ............................................. 8  
  2.1.1 Easy to Use the Interface ............................... 8  
  2.1.2 Saving battery life ..................................... 8  
  2.1.3 Reliable GPS-data ...................................... 9  
  2.1.4 Handle False Information .............................. 11  
2.2 The Correctness of Camera Placement ......................... 12  
  2.2.1 Graham Scan .......................................... 13  
  2.2.2 Running Time .......................................... 14  
2.3 Client and Server Communication ............................ 14  

### CHAPTER 3 – EVALUATION

3.1 User Feedback from Test Users ............................... 15  
  3.1.1 Easy to Use the Interface ............................... 15  
  3.1.2 Battery Life ............................................. 16  
3.2 Camera Placement ............................................. 16
CHAPTER 1

Introduction

This thesis is about geographical based wikis which use the combined knowledge of many to gather information. It is different from sensor and other measurement dependent systems in the sense that it relies on its users to collect data. The work in this thesis revolves around a proof of concept prototype which is a speed-camera detection system built on the geographical based wiki principle. This system takes advantage of the fact that a lot of people travels on our roads and offer drivers the opportunity to help each other out. The reason that speed-cameras has been chosen for the proof of concept prototype is that you can find all of the stationary speed camera positions in Sweden on Trafikverkets homepage [1]. Therefore the data gathered in the application can be verified against Trafikverkets positions.

1.1 Background

There are basically two types of mobile applications, those that can operate on their own and those that need the ability to send or receive data. In the first category is for example a simple game like Tetris. But by simply adding the possibility to have a global high score list will change this simple game into the other category of applications which have the need to send and receive data. However other types of information are not as easy to gather as fetching the score from a game and sending it to a server. In many cases sensors or other information gathering devices could be used to feed users the information they need. However sensors are not that cheap and if we would like to know for example if there are any animals on the road. We would have to deploy a lot of sensors, which in turn would cost a lot of money to buy, deploy and maintain. Therefore using real people as an information gathering system would be more convenient and a lot cheaper. However this type of system needs quite a lot of input to work properly since we can not simply trust a single user to register all the necessary points. The system should use the knowledge of many combined or what you could call group awareness.
1.2 Goal

The goal of this thesis is to investigate the necessities needed to create a location based wiki. And to develop a proof of concept prototype which is based on this principle. The system should use crowd computing or group awareness to locate speed-cameras. This is done by using the users as the information source in the application. Speed-camera data should be stored on a remote cloud server backend. There should be a possibility to store additional data connected to the different locations to determine in which lane the speed-camera is monitoring you. The application should support mobile speed-cameras as well, since this is not available in applications with preconfigured speed-camera locations such as dedicated GPS-units [2]. This will show end-users the real potential of the system.

1.3 Delimitations

To define where the focus of the work done in this thesis will be, delimitations have to be set up. The security of the system will not be taken into account. Searching through the database could become a quite heavy operation if other more common positions then speed-cameras are registered, or if the application was to be used worldwide. However this will not be investigated in this thesis. Only a proof of concept prototype will be built and therefore no finished commercial product.

1.3.1 Requirements

Building the client part of the system on an iPhone was a requirement of Neava, the company which the application was built at and not an active choice of the author of this thesis. The proof of concept prototype is meant to be used by drivers inside of a moving vehicle. Therefore a requirement is that the interface should be easy to use for people with very little smartphone or computer knowledge. Since the application is meant to be used while driving, it should only require a single button press and should be easy to operate. Since the application’s information gathering mechanism is its users there should be a way to handle false information. There is always a possibility that users accidentally pushes a button or simply wants to break the system by registering faulty cameras. This type of false information should if possible be handled in the application. Since the application is built upon group awareness, the server side should have the ability to merge several registrations of the same camera into a single one. In such a way that it represents the real physical camera in the best way possible. There is a possibility that outliers could be present in the system and they should be dealt with in such a way that one or several outliers will not drastically change the end result.
1.4 Key Concepts

The proof of concept prototype is built upon merging several different technologies together. The system uses the common knowledge of many, group awareness or crowd computing to gather information, which are all different words to describe the same thing. It means that if you ask a question to a large group of people and trust what the majority of those people say, you will probably get quite close to the real answer. The application uses cloud computing to store its data on a remote location where it can be processed by the cloud server and the processed data can be read by other users. Cloud computing basically means that you do not need to have a lot of storage space and computing power on your workstation or in this case on your mobile phone, since the calculations is actually performed in the cloud instead of in your phone. The geo-tagging of information is necessary in the proof of concept prototype to determine where the speed-cameras are located and to download information which is relevant to your current location.

1.4.1 Cloud Computing

The expression cloud computing is a popular term to use today. It is an internet based service which basically allows you to buy computing power or storage space on demand instead of buying a high-end workstation or server see figure [1.1]. For you as a customer, this means that you do not have to worry about maintenance and that you will only pay for the resources that you actually use. For the cloud service providers, this means that when you are not consuming the resources they can hire it out to someone else.

1.4.2 Crowd Computing

Crowd computing is another of those up and coming terms which basically means that we all help out with information gathering. Take for example a large group of people standing in the street when you are driving by, not knowing the way into town. When you ask them for the way they will all point in a different direction. Someone would probably point in the wrong way and others would point approximately right. Going in the direction where the most people point will probably get you into town. This is achieved by taking the combined knowledge of the "crowd" to solve your problem. More computerized examples of crowd computing include social networks, blogs and wikis.

1.4.3 Geographical Positioning Wiki

Combining cloud and crowd computing would mean that people work together by sharing information. The information gathered is stored in the common storage space where they all have access to the information. To this add the possibility to record your position and you have a location based wiki.
1.5 Exploitation

A geographical based wiki could be used for so much more than a speed-camera detection system. Basically it could be used for anything where people benefit from working together with geo-tagged information. In the subchapters 1.5.1 and 1.5.2 two different examples of geographical based wikis will be explained in short. Where the speed-camera prototype can be seen as a part of the Road Assistant explained in 1.5.2.

1.5.1 Travel Guide

Imagine that you are travelling to New York City. Instead of buying one of those travel guides and read up on stuff to do on beforehand you jump on the plane directly. On arrival you start an application where others have registered sites worth visiting combined with geographical coordinates, so that you can display them on a map. This will make life easier for you as a tourist since there is no need to study before your vacation and you will get the combined resume from all users not just from the author of a book. In
order to further improve the application you should contribute with some information
yourself.

1.5.2 Road Assistant

Traveling by car is a part of everyday life for a lot of people. Many spend a few hours each
week on our roads. Therefore, a road assistant that helps you keep track of icy roads or
wild animals, warn you for sharp turns and road construction sites might be helpful. To
get these warnings you will rely on other drivers registering points that you will benefit
from. The only thing you have to do in return is to register some points yourself that
could be helpful for other drivers. This is a good example where crowd/cloud computing
combined with a geographical positioning system could do a lot of good, make our roads
safer and even save lives.

1.6 Related Work

1.6.1 GeoPedia

Geopedia [3] is a location based wiki. It will show you wiki articles based on your current
position. The usefulness of this application is quite easy to comprehend. If you are in
an unfamiliar city and have some time to kill, you could use this application to find a
place to visit, something to do or information about the city you are in. Basically it
could be used as a traveling wiki mentioned earlier in this paper. The difference between
this solution and the speed-camera detection system developed in this thesis is that all
kinds of information can be stored within this application and it has therefore no specific
purpose.

1.6.2 LBWiki

LBWiki [4] is another location based wiki that allows you to create nodes. These nodes
could be a geographical position or an area. One thing that is different about their work
compared to GeoPedia [3] is that they use a hierarchical approach. You could specify
Europe as a node then under that you specify Scandinavia as a child node to Europe.
Then you specify Sweden as a child node to Scandinavia and so forth. This way it is
possible to add multiple nodes that overlap each other or lies completely within each
other as in the example above. This means that you have the possibility to choose
at which hierarchy level you want the information. This principle could be used when
further developing the speed-camera detection system into a complete road assistant. For
example a speed-camera could be a child-node to an icy road if it is located within its
covering area.
1.6.3 Blackberry and iPhone CoDriver Application

The Blackberry and iPhone has a CoDriver application available in Australia and New Zealand. This application will act as a co-driver for you. It will warn you for stationary speed cameras, railway crossings, accident black spots and much more. However it does not really detect anything on the road at all. Instead they use a local database on the phone with on beforehand saved positions. These positions are actually visited and recorded by specialized field based staff members. Even though they want feedback from users, each tip on a faulty position is visited and updated by a staff member. This application has basically the same functionality as the speed-camera detection system developed in this thesis which could be extended to cover other points as well. However the application developed in this thesis provides this type of information in a more cost efficient way since there is no need for staff members to visit each location in order to register it. The application rather relies on its users to provide the necessary information.

1.6.4 Road Sign Detection and Recognition

Mark Schneider has presented research on how to detect road signs by a video camera mounted on the roof of a moving vehicle. This was done by detecting colors and shapes in the images. The sign detection algorithm developed was working but had a few flaws. It was sensitive to break lights from vehicles in front of the test car and resulted in a lot of false detections. During the tests it was found that forsythia bushes had a color similar to that of warning signs and a bunch of false detections were due to this coincidental color match. Worth mentioning is that some road signs were not detected at all. As mentioned in road signs are designed and developed to be easy for humans to perceive. But it is a challenge to enable computers to recognize them. This system has the same purpose as the speed-camera detection system developed in this thesis with the difference that it targets road signs rather then speed cameras. During tests of the road-sign detection system it has been shown that this type of application is not that reliable. The speed-camera detection system on the other hand relies on human vision to find and perceive the position of the speed-cameras and not real-time image analysis software.
Development of this system needs a few basic building blocks, see figure 2.1. First off a mobile device is needed, or in this case an iPhone with a GPS receiver and the ability to download and upload data to the cloud-server. The server should have the ability to process and store the data.

The GPS-receiver will fetch the iPhone’s position (1). With every location found a few tests has to be made. First off validation of the location received, by making sure that it is accurate enough. After that the system should check if there are any cameras nearby.
and display that information to the user.

Whenever the user decides to register a new camera, which is represented by the current position and some information to register which side of the road the camera is located on should be sent to the server (2). On the server side the data is processed, what the server should be looking for is if there are any other points registered that could represent the same camera. Then the server performs an analysis of that data to see where it is most likely that the real physical camera is located. The result should be a best effort estimation of the position given the dataset the server had available at that point in time. The result is saved in a database (3) where all the client machines should have access to download the current camera locations.

### 2.1 Important Features

#### 2.1.1 Easy to Use the Interface

The application is meant to be used and operated inside a moving vehicle. This requires an easy to use interface, see figure[2.2], so it will not interfere with your driving. An easy user interface means that persons with very limited smartphone and computer experience should be able to use this application after a brief introduction. The buttons should be big enough so that they can be easily found and pushed while driving. All the necessary operations should only require one tap on the phone. Any auto locking mechanism should be disabled if possible to ensure that your attention stays on your driving.

#### 2.1.2 Saving battery life

If not one of the more critical features of the system. Saving battery life on the client could improve the usability of the application. It is not deemed critical since the application is meant to be used in a car where a 12V socket often is available. Enhancing battery life could be done by letting the GPS work less. This will affect the precision of the locations and a trade-off between location precision and battery life has to be made. Another battery saving feature could be to turn off or dim down the display. The application will work in the background most of the time and the display does not have to be on as long as there are no cameras nearby. When a camera is found in the proximity the display could flash or the phone could simply use a warning sound instead. The display should be turned on if someone puts a hand near the phone or taps the display.

Changing the brightness of the display was not possible since Apple do not allow you to tamper with the brightness of the display in order to allow the application trough App Store. Changing the workload of the positioning system was not an option either since the accuracy was worse then 30 meters when turning with the accuracy set to the best possible. The accuracy of the application should not suffer in order to enhance battery-life.
2.1. Important Features

2.1.3 Reliable GPS-data

The GPS data is dependent on how good the iPhone is to find your location. The iPhone uses a framework called Core Location to find where you are \footnote{[8]}. There are three technologies that Core Location uses in order to do this, GPS, Mobile Positioning System (MPS) and Wi-Fi positioning service (WPS). GPS which is the most accurate reads microwave signals from satellites to determine your location. The second best option is MPS. It does a calculation based on the cell towers which are in range to determine your position. This type of technology can be quite accurate in areas with high cell tower density. The third and last option that Core Location uses is WPS, it uses the IP-address from the iPhones Wi-Fi connection and tries to guess your location by using a database of known service providers and the areas where they service. WPS is not particularly accurate and can be off by many kilometers. In the speed-camera detection
system the GPS makes the most sense to use since it is the most accurate of the three and the technology is in general available where speed cameras are located. However it is not possible to tell the iPhone which technology to use. What one can do though is specifying how accurate the phone should be and it will choose which technology to use accordingly [8]. There is no guarantee that the accuracy specified will be delivered so it is necessary to keep track of the accuracy and warn the user if it is too far off. In order to determine in which lane the camera is positioned the application makes use of the current direction and match this with the direction the user had when registering the camera, see figure 2.3.

Therefore the system should not allow users to register cameras without a proper direction given. Getting the direction the iPhone is heading in is quite simple since it is already done for us when using the Core Location framework. When matching the direction a margin of error has to be used since the direction can differ when driving and the GPS positions has a margin of error as well. In the proof of concept prototype a margin of error up to 45 degrees on each side of the direction given has been used. Such a big span is used to avoid warning delays in sharp turns. And still avoid faulty warnings, in for example a four-legged intersection.
2.1. Important Features

2.1.4 Handle False Information

The application is a crowd based application that is dependent on user input. Using this approach means that there is a possibility that users might register cameras where there are no cameras. This can be due to errors where the user accidentally presses the button, or the user tries to break the application. There is also the possibility that a camera was recently moved or removed from the location.

In order to deal with these erroneous camera positions there are a few choices to choose from. One possibility is to use statistical methods in order to find places where there most likely would be a camera. Problems with this method would be that cameras in remote locations will probably not be registered as often as cameras on for example freeways and they could therefore be mistaken for false camera placements. Another problem would be if a camera moves and there are a high number of registered points at that particular location. The system would still think that the camera is there. A solution to this would be to force users to register cameras at a certain frequency in order to keep the camera alive. This could annoy users that might feel that it is unnecessary to register a camera that already has a good position.

Another approach would be to try and find users that register a lot of false positions and not allowing them to use the application. This could work fine in theory but a problem is sorting them out from a user that accidentally pushes the button. In order to allow accidental pushes but not persons that tries to break the application, a lot of registrations has to be done in order to see a pattern. This will make the application malfunctioning for a while before the bully is caught and all his registrations could be removed by the system. In order to deal with those accidental pushes a registration alone can not make a downloadable camera position. Instead the same position has to be registered by multiple users in order to be confirmed as a real position.

To allow one registration to be regarded as a true camera position a "report false cameras" button could be used where the system will remove the corresponding camera position after a number of false reports. This would also take care of the problem where a number of people have accidentally reported a position where there is no camera.

In a real commercial product a combination between excluding users that misbehave and a reporting system would be a good idea. It would give users the opportunity to take care of the faulty camera positions immediately and users that continue to misbehave will be taken care of by the system. But in the proof of concept version only the report false camera button has been implemented since there will be a very limited number of test users and the tests will therefore mainly be focused on how the system works when used in the intended way. By allowing a single registration to be considered as a real camera position, the test users will quickly see the results of their registrations and probably enhance their initial experience of the system.
2.2 The Correctness of Camera Placement

Since the application developed is a crowd based application, all data has to be combined into a single point. The simplest way to do this is to take the mean value. The problem with this solution is that outliers will have a big impact on the end result. Since there will be no opportunity to study lots of in-data with a lot of users and adapt a mathematical model after this, every point should affect the end result in a similar manner. Another issue that has to be taken into account is the time it takes for the algorithm to run. In this proof of concept version with a small number of users this will not be a concern. But when using this in a commercial product with a lot of users this will most definitely become an issue.

When choosing an algorithm to use, the choice fell upon convex-hull peeling. This is due to the fact that an outlier will contribute to the end result as much as any other point. The algorithm will calculate a convex-hull which is the convex polygon that will include all points in the dataset, see figure 2.4. Imagine nails that stick up from a piece of wood. Over all of them you put a rubber band. The nails that have contact with the rubber band are part of the convex-hull.

Then the points that belong to the convex-hull will be removed and the algorithm will run again and again removing one hull at a time until it can not calculate any more convex-hulls. At this point you should have one or a few points left which is considered to be the points in the middle of the set. If there are more then one you could use the mean value of those points in order to get your camera position. By using this type of algorithm you will get the center point of the group and outliers will not affect the end result any more then a point closer to the middle of the set.

![Figure 2.4: The convex-hull of the dataset](image-url)
2.2.1 Graham Scan

The method that has been used to calculate the convex-hull is called Graham scan [9]. In order to use the Graham scan you need to sort the points with the smallest x value first. If there are two or more points which have identical x values you should sort them with the smallest y value first. Then by using three points each time, you calculate if the points are on a straight line or if it is a right or a left turn. If it is a right turn the middle point of the three can be considered to be a part of the hull, but if it is a left turn or a straight line the middle of the three points is no longer considered as part of the hull and can be removed. Each time a left turn is found you should backtrack until a right turn is found removing the middle point each time. This should be repeated until you come to the end of the list at that point you will have the upper part of the hull. In order to get the lower part you start again but from the end of the list moving towards the beginning, see appendix A for pseudo code. There is a flaw in the pseudo code that has to be taken care of. It only tells you to sort by x-coordinate, however the algorithm requires that you sort first by x-coordinate and then by y-coordinate in order to avoid errors. This is illustrated in, figure 2.5 In order to calculate if it is a right or a left turn, see appendix B.

![Figure 2.5: Difference of the right-turn calculations when sorted in different ways](image)
2.2.2 Running Time

Before starting to calculate the hull the points need to be sorted as previously stated in section 2.2.1. Sorting a dataset can be done in $O(n \log n)$ time [10] by using for example Mergesort. Even though Quicksort has the worst case of $O(n^2)$ it is often faster than Mergesort, since the partitioning in Quicksort can be done significantly faster than the merging in Mergesort [10]. The proof of concept prototype uses the ORDER BY query in MySQL [11] to sort the data, which uses a mixture of the two. 

In order to calculate the upper hull the algorithm uses a for loop with a while loop inside. The for loop is run a linear number of times since it will work its way through all the points in the dataset. The while loop is executed at least once in each run of the for loop, any additional runs will remove a point that can be considered not belonging to the hull. This point will not be checked again in calculating the upper hull. Therefore the total number of times the while loop is run, including all the runs through the for loop, is bounded by the number of points in the set [9]. The lower hull is calculated in a similar manner and the execution time will therefore be $O(n)$ since any constant before $n$ will be discarded. This means that speed of the algorithm is dependent on the speed of the sorting algorithm.

2.3 Client and Server Communication

In order to communicate between the client and the cloud-server the client simply opens a webpage to download camera information. The client will open a new webpage each time a point is registered or new data is downloaded. All the functionalities in the client correspond to a different page on the server. The pages will contain a PHP script which opens up and gets the data or saves the data to the MySQL database depending on the functionality needed. The data stored in the database is the camera coordinates and the direction it is positioned in. By using this method the system could be extended in a simple way by just adding a new script on the server and a corresponding button to open the correct URL on the client. However no web pages are shown as such but rather the information on the webpage is downloaded and used in the client application. This opens up the possibility to use the same backend, on for example an android application.
CHAPTER 3

Evaluation

3.1 User Feedback from Test Users

The application has been tested together with professional drivers at DHL in order to get test results from a real user environment. After a short introduction they were given an iPhone with the application installed and instructed to use it in the best way they could. After a period of time the results were evaluated and compared to the important features stated in the System design chapter.

3.1.1 Easy to Use the Interface

One of the requirements of the application was that it should be simple to use and operate inside of a moving vehicle. After the first test run the overall impression was good but a few adjustments were necessary. They thought the screen was too bright during night time travel. One of the battery-saving features was actually to dim down the display. However Apple has its own functionality to dim down the display dependent upon the ambient light in the room but this was unfortunately not enough. Apple will not allow developers to tamper with the brightness of the display. There are a few workarounds to somewhat simulate this. One is putting a black semi-transparent image on top of the view which simulates a dimmed down display. The other solution is to use the proximity sensor which is used to turn off the display when you are talking in the phone. The positive thing with this solution is that it actually turns off the display and saves battery-life. The negative thing about it is that you need to turn the phone upside down or otherwise cover the proximity sensor which is situated on the front of the phone. Another issue related to the user interface is that the test users wanted to know when registration of cameras was successful rather then unsuccessful, which was previously the case.
3.1.2 Battery Life

The battery saving feature had two options on power saving. The first one was to dim down or turn off the display. Unfortunately this was not an option since it is not allowed by Apple as mentioned in the previous section about "Easy to use the interface".

The other option was to change the workload of the GPS. On the iPhone there is an option to specify how accurate position you want. However there is no guarantee that the required accuracy is met. Developers could see it as a desired accuracy. Even with the desired accuracy set to best possible the positioning system in the phone struggled when the direction was changing quickly. A tradeoff between battery life and reliability of the application had to be made. In this case the reliability was ranked higher and therefore the GPS accuracy was set to best possible.

The battery life was measured by the test-users and reported to be about six hours. This was measured during a normal truck-drivers day and included stops for loading and unloading cargo. During these times the workload of the GPS was significantly smaller.

3.2 Camera Placement

The camera placement is one of the absolute key features and something that can be extremely useful in further development of geographical based wikis and further development of the road assistant. In the subsections 3.2.1 and 3.2.2 conclusions about the real measured result and how the users perceived the quality of the positions is highlighted. Some surprising results with the camera positions are mentioned as well.

3.2.1 Difference in the Result

The difference in the results between using the convex-hull peeling and the mean value was not that big in the real user test, only a couple of meters in the average case. This is due to the fact that the test users had been very precise when registering points. Since in most cases the registered points were very close to each other. The convex-hull peeling algorithm is really used to reduce the impact of outliers but will have a result that looks a lot like the mean value when the points are evenly distributed.

It was noticed when analyzing the result of the user test is that the registered cameras from the test users were always placed before the actual camera positions from Trafikverket. What could be seen in tests of the GPS on the iPhone was that the coordinates from the iPhone was a bit behind the actual position when the test subject was on the move.

3.2.2 User Experience

According to the test-users the warnings came on time and the warning system was easy to hear and understand. When comparing the test results between the mean value
and the convex-hull peeling algorithm. The test users mentioned that when deliberately registering a camera really late, the warning was sometimes a bit delayed when using the mean value. The test users stated that in both cases they got the warning well in time and that there was no difference in the result between the two methods, when registrations were done with a best effort attempt.
The goal with this thesis was to evaluate the necessities needed to create a geographical based wiki as well as building a proof of concept prototype with this principle which should warn users of speed-cameras. One of the requirements of the application was that it should use group awareness to find the information it needs. This has been done by allowing users to register speed-cameras themselves. No sensors or other measurement system has been used for this task. An important goal with the application was that the data should be stored on a cloud-server backend to allow other users to access the data. This has been achieved by using a simple web-server and a MySQL database. These are quite simple means to create something that can be quite powerful. One of the drawbacks with this kind of system though is that it needs to motivate its users to actively register cameras.

The goals in this thesis have been fulfilled since a working prototype has been developed that uses crowd and cloud computing to find and save information. Furthermore additional data connected to the geographical position is being stored. This data is in the proof of concept, prototype the direction in which the iPhone traveled when registering the camera. The information is needed to determine in which lane the speed-camera is monitoring you. Mobile speed cameras have been included in the application as well.

4.1 Using Crowd Computing

Using group awareness or cloud computing has its drawbacks. There is a possibility that users accidentally pushes a button or that users try to break the application by registering faulty cameras. This needs to be handled in some way. There were a few methods mentioned earlier in the "Handle false information" section of this thesis. The possibilities were to allow users to report a faulty camera position or to use statistical methods to
find where there is a high probability that a speed-camera is located dependent upon all registrations sent to the server. The last option mentioned was to prohibit misbehaving users from using the application.

All of the options listed above have their own flaws, allowing users to report faulty camera positions will require users to act when they come across a faulty camera position. This might be ok since it is a crowd based application where the users are in control of the information, since they are the information providers themselves. But there is always a risk that some users will report correct camera placements as faulty in an attempt to break the application. To have some protection against this, the application could use a unique id from each phone to ensure that a number of different users have to registered the same camera as faulty before the camera is removed. However it could become difficult to determine how many registrations that should be necessary to actually remove a camera. Should it be dependent on the total number of users? Should it be a fixed number? Or perhaps it should be dependent on the number of unique users that have registered that particular camera? All of these methods have their own drawbacks. For example using a fixed number of registrations could mean that very few "false camera" registrations could over-ride a big number of camera registrations. Or on the other hand a limited number of registrations have been made on a camera, and it needs lots of "false camera" registrations for the camera to be removed. If the system used a percentage of the number of total phones that uses this application it would mean that no account has been taken to how many users are using the application in that particular area. If the system would take this into account a percentage of the number of registrations on that particular camera could be used. If this system were to be used it would not be very responsive if a camera that had been at a certain place for a long time was moved. Other alternatives to remove faulty cameras were to use statistical methods to see where there is a high probability that a camera is located. The risk with this type of system is that cameras in remote locations could be missed and seen as a faulty registration. The last option was to ban misbehaving users, but this requires the application to make the distinction of accidental pushes from misbehaving users. In order to not risking users to be banned from an accidental push quite a few faulty registrations would have to be made before a user can be removed. And this still does not answer the question what can be seen as a faulty registration.

There is no perfect way to solve this problem for all types of applications. It should rather be adapted to each application dependent on the behavior and amount of users. In a real commercial product a mixture of several different methods could be used. For example a report false camera position button together with a behavior analysis of each user which will warn and ban users after some time if they are misbehaving. In the proof of concept prototype, which is a very small test application. Only the report false camera alternative was implemented.
4.2 **Using Cloud Computing**

Using cloud computing seems like a perfect solution on a mobile phone which has limited computational power, memory and battery-life. Allowing the "cloud" to store its data and perform some of the calculations is in general a good idea, but this idea has its own drawbacks. You need an internet connection available at all times for the application to work properly since it needs to download new data. Using data traffic is not cheap and a data plan would be a good idea to have for end-users of this type of application. This extra cost might scare some users away from the application.

4.3 **Testing the Application**

When testing the application only a very limited group was used. Therefore the result might have looked differently with a higher number of test-users with a bigger diversity. For example there were no female test-users. The application was only tested in the northern part of Sweden within a very limited time. Mainly by truck-drivers which knew the camera-placements by heart on beforehand. Due to this fact it can be difficult to draw any strong conclusions about how this type of application would have been met by the public.

4.3.1 **Battery-life**

The battery life of the application was tested during a truck-drivers normal day and not continuous driving and could therefore differ quite a bit. Since the application is meant to be operated inside of a vehicle where a 12V socket usually exists this is not considered a huge problem. The truck-drivers reported battery-life to about six hours which should be enough for commuters to use this application to and from work without being forced to buy a car-charger kit.

4.3.2 **Small Difference in the Result of the Camera Placements**

All the camera-registrations were very close together. This could be due to the fact that test-users mostly knew were the cameras were located on beforehand, and they were very thorough when registering new cameras. This in turn meant basically no outliers. The algorithm used made the application less sensitive to outlier data and in this small test the system could have used the mean value instead and still kept the user experience at the same level. But if this application would have been released to a larger group with a larger diversity we would perhaps have gotten a different result. Testing the user-experience of the camera positions was done by looking at the difference in the perceived quality between simply using the mean-value and the convex-hull peeling algorithm. When comparing the two there was no difference in the user-experience when
doing best effort runs. When the test-users were told to deliberately register cameras late they mentioned that warnings sometimes came a bit later but still on time when using the mean value. This shows that getting the camera position exactly right is not that important since the warning should still alert the user a few seconds before the actual camera placement. Adapting how early the warning is issued depending on the speed of the user could improve the user-experience.

4.4 Relying on Users

The application relies entirely on its users. Without their co-operation the system won’t work since it is based on their input. It is a "give and take" system where you are supposed to register some positions in order to help others and in return they will help you. The problem with this type of application is to motivate users to actually register positions. Motivating users could be done in different ways for example having a contest or give users a discount if they fulfill a certain registration quota. Professional drivers could be motivated to use this application in order to help each other out in the traffic. However it could be difficult to motivate users to register cameras without them seeing a personal benefit from it. It will be particularly difficult to convince the very first users of the application since they do not have a large database to get information from. In order to enhance their experience a few things could be done. First of the stationary speed cameras from Trafikverkets homepage could be inserted into the database on release. Perhaps involving Trafikverket further into a pilot project where road construction sites are registered by on site personnel. Motoring news on the radio could be inserted into the database by the radio operator. These are a few of the arguments that could convince users to start using the application.

4.5 Future Work

A nice feature that would really improve the usability of the proof of concept prototype is having voice control instead of using button presses to register points. This would improve the possibilities of the number of different types of points to register as well. Since the requirements has previously stated that no more then a single tap on the phone should be necessary to perform all tasks needed to operate the application in a moving vehicle. The limiting factor would otherwise be the number of decent sized buttons that could fit into a single view. Using voice control would maximize the possibilities of the application and would further improve the safety of the application since no physical interaction with the phone when driving would be necessary.
Algorithm ConvexHull(P)
Input. A set P of points in the plane
Output. A list containing the vertices of the Convex hull CH(P)
Sort the points by x-coordinate,
resulting in a sequence p_1, p_2, . . ., p_n.
Put the points p_1 and p_2 in a list Lupper, with p_1 as the first point.
for i ← 3 to n
    do Append pi to Lupper.
while Lupper contains more than two points and the last three points
    in Lupper do not make a right turn
        do Delete the middle of the last three points from Lupper.
Put the points p_n and p_{n−1} in a list Llower, with p_n as the first point.
for i ← n − 2 downto 1
    do Append pi to Llower
        while Llower contains more than 2 points
            and the last three points
            in Llower do not make a right turn
                do Delete the middle of the last three points
                from Llower.
Remove the first and the last point from Llower
to avoid duplication of the points where the upper and lower hull meet.
Append Llower to Lupper, and call the resulting list L.
Return L
Calculation of right or left turn

FUNCTION Cross_product(p1, p2, p3)
    RETURN (p2.x - p1.x)*(p3.y - p1.y) - (p3.x - p1.x)*(p2.y - p1.y);
ENDFUNCTION
Bibliography


