UNIVERSITY OF AGDER

Project report in IKT420

Crisis Mapping

Use of Gaussian processes to estimate fire spread

A project report by

Håkon Bakkevig Steinsholt
Enok K. Eskeland
Leonard Loland
Daniel Aasen
Charika Samangi Perera Kukulage
Mahsa Hassankashi

Supervisors

Ole-Christoffer Granmo

This project is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education.

University of Agder, 2011

Abstract

Crisis mapping is a concept to help crisis management teams to determine the best course of action in a changing or evolving crisis situation in order to help people and save more lives. This report describes how mobile phone sensors can be used to measure, interpret and visualize a fire crisis situation. An interactive simulator was especially designed for this project to provide sensor readings affected by environmental parameters such as wind and humidity. These individual readings were used in the Gaussian Processes Regression (GPR) to estimate the area and magnitude of the fire. The kernel function of the GPR was expanded to also take wind into account to the distance vectors.. The overall system performance were visualized by putting the actual situation from the simulator together with the estimated situation from the GPR. The final results of this work shows acceptable accuracy and serve as a proof-of-concept that mobile phones can be used to map fire crisis situation.

Contents

List of Figures

List of Tables

Definition list

Chapter 1

Introduction

1.1 Background

Crisis Mapping is the process of collecting, analyzing and visualizing crisis data for further crisis management. The goal is to provide better real time decision support to determine the best course of action in a changing or evolving crisis environment. This in turn provide help to crisis victims and save more lives.

In 2010, a major earthquake crisis struck Haiti. The shock measured 7.0 on the moment magnitude scale (MMS) and was followed by another 52 aftershocks of 4.0 MMS or higher over 12 days. 1 517 000 people and their dependents were affected. People were in need of housing, food and medical attention. Mapping these needs would improve further crisis management [].

The needs on Haiti were mapped by Ushahidi, an open source software. This software collects, analyzes and visualizes data from social media, text messages and more. The Haiti government took advantage of this software and allowed crisis victims and bystanders to report needs to temporary web services and text message services. These services together with Ushahidi provided a crisis map that aided the government to send help where it was needed [].

Today, researchers are investigating how to use mobile phone sensors to map crisis situations. Many mobile phones have built-in sensors such as microphones, accelerometers, gyroscopes, barometers and GPS. In theory, these sensors can be used to detect earthquakes, fires, explosions, collisions and other crisis situations by reading abnormal magnitudes or patterns in sound, acceleration, air pressure and temperature. Mobile phones can be used to report this abnormality to a central crisis mapping system and possibly aid further crisis management.

1.2 Problem Statement

CISTECH of the University of Agder proposed a basic project about crisis mapping using mobile phones. Because crisis mapping is a general term and includes a broad range of different crisis situations, we had to narrow down the goal of the project. After consulting with our supervisor, the final goal was to create a prototype crisis mapping system that could detect fire using mobile phone sensors. We chose fire in specific because it can change or evolve over time. However, the chosen topic does not come without a set of practical limitations.

Given the project goal, we focused on two main problems that had to be solved. Firstly, fires had to be simulated. There was no way to justify starting real fires to gather this data. Simulating fire is challenging because fire spreads based several factors such as wind, heat, humidity and other over time. Secondly, mobile phone sensors do only report readings at their location and do not cover burning areas without nearby sensors. Uncovered areas have to be guesstimated.

1.3 Literature Review

1.3.1 A Survey of Mobile Phone Sensing

The field of mobile phone sensing is rapidly expanding. The accelerometer in a smart phone can determine whether the user is walking, running or standing still. The microphone can collect audio. The light sensors can determine whether the phone is in a pocket or not and the barometer can determine air pressure. In addition data can be combined from different sensors to make assumptions, for instance a phone can sense that a user is walking down a hill and in which direction he is going. Specialized sensors have also been used to measure air pollution and could perhaps be used to detect explosive residue.

There are many issues with properly utilizing and gathering data. First of all a large amount of users to properly test a system is needed. This can be solved by distributing the application via the App stores and providing an incentive for the users to use it. Second there are huge privacy issues with listening in on private conversations using the microphone and tracking the user via the GPS. This can somewhat be fixed by calculating any result on the phone and sending only the results, however this would require more resources from the phone and would likely limit battery time or performance. Some studies have shown that sensing systems have reduced the standby time by significant amounts, vastly affecting

the user experience.

Creating a system that produce accurate results may also be slightly problematic as vendors did not foresee the use of their sensors in new applications. The sensor sampling rate varies based on the CPU load and the API and OS does not support complete access to the sensors. However, as the mobile phone sensing field is growing vendors are likely to catch on and provide better sensors and sensor support in future releases.

Moreover, it is also difficult to create a program that fully accounts for all possible instances. Explosives used in demolition and terrorism both create the same or similar sound and gun fire while hunting sounds the same as gun fire in a super marked. A phone also needs to know if it is currently inside a container where the sound might be suppressed, in which case the system might become more sensitive to sound. This in turn can be somewhat solved by using light or proximity sensors however there might be cases where the phone is placed in an unusual location and therefore provides inaccurate results.

1.3.2 Distributed Perception Networks for Crisis Management

Distributed Perception Networks for Crisis management (2006) focuses on the fusion of information gathered from fixed sensors and human perception. The fixed sensors are few and scarcely distributed in an area where a crisis could have devastating consequences. Endsley's model tells that first comes perception, then comprehension and projection. The paper focuses on combining the perception of humans and the fixed sensors to create a clear picture of the situation. For example when the sensors are picking up dangerous levels of ammonia in the air the system will use human perception to gather more input. This is done by sending SMS to everyone in that area where the query could be; "Do you smell ammonia?". By using human perception the crisis management team will get a richer and more detailed map instead of only using the scarcely distributed fixed sensors.

Sensor readings and public perception results in a vast amount of information which needs to be processed. Since it is such an enormous task for manual human labour the paper proposes to use a Bayesian network to find the most probable cause of the crisis. There is a large degree of uncertainty when gathering huge amounts of information from people and possibly faulty sensors. The DPN architechture represents a multi agent system which means it can comprehend large problems in efficient manner. A set of nodes is processed by one agent before they send it to their father node. The paper does not suggest how to calculate how toxic the air will become where it will spread.

1.3.3 Video about how crisis mapping helped during the Haiti earthquake

When the earthquake hit Haiti in 2010, there was an initiative to map where help was needed to increase the efficiency of rescue personnel. In addition, no emergency call centers were operational directly after the earthquake, further increasing the value of a crisis map in this particular situation. People in need of help would send a text message signifying their location and situation, operators would then note what help was needed and place the note on the map showing the location. Organisations could then use this map to deploy personnel in areas where they were needed the most. In addition, any volunteers could view the map and help if they were able.

One thing to note about this system was that messages were written in different languages and then translated to English. Most of the translation were done with assistance from the general public, this highlighted the positives of a crisis mapping system that is open to the public.

By using a map and plot what is happening in the different areas, organisations that are trying to help may send their resources to the correct locations at the right time. Organisations may also inform people where dangerous locations are during the crisis and guide the people in that area to a safe location. In addition, they may plan their next move as they observe the evolving crisis. People that also are able to see this crisis map may see the locations that are dangerous, for example they can then use the map to spot a burning building and head in a different direction.

1.3.4 A Real-time Disaster Situation Mapping System for University Campuses

A real-time disaster situation mapping (RDSM) system is a web-based system which acts as a social media. The system helps its users and the disaster management team during a crisis situation by giving the disaster management team a clear overview of the situation and the users can use the system to acquire their location and the nearest exit.

The RDSM system is based on some conditions such as disaster type, place of use and targeted users. The main focus of the system is to protect people during and immediately after a disaster based on the assumptions that the earthquake has a minimal impact on the university buildings. In the event that the university buildings are severely damaged then important equipment such as servers, Wireless LAN, etc. may be out of commission and the system may end up not working

properly. The mobile devices, such as computers or phones, are assumed to be in use and connected to the servers during the disaster situation. The RDSM system can be accessed inside the university as well as in the evacuation office by any employee with sufficiently high authority level.

The RDSM system has two major parts, called situation gathering sub system (SGS) and the situation mapping sub system (SMS) []. The SGS allows people in the campus to send situation information about their location, such as degree of longitude, latitude and name of the place at the time of the disaster, using their mobile devices.The location information can be identified by manual entry as well as by the automatic acquisition using Place Engine. Place Engine is a service where users can send location information, even though they do not recognize their locations, by utilizing Wi-Fi devices. Some requirements have to be satisfied in order to set up the Place Engine for automatic data acquisition of location information. First, the Wireless LAN access point data must be saved in the Place Engine database. Secondly, at least one access point must be reachable from the users location. Next, access to the Place Engine server must be feasible. Finally, the users' mobile device(s) (PC or phone) must be equipped with Wireless LAN (Wi-Fi) equipment and the client software must be installed on those mobile device(s) []. The users can send situation information from the situation data entry screen by accessing the SGS. Speech to text transformation is achieved by using "w3voice", which is a development kit for voice-enabled web applications []. Therefore the informers can send voice data, text data as well as video via the disaster situation entry screen.The entered situation data is stored in the situation database.

The SMS reads data from the situation data base updated by SGS and it will replicate that data onto maps sent by the users. The more detailed information of the disaster situation is shown by the time series on the map screen. The users can also access the voice or image data in order to get a clear and accurate information about the crisis situation.Situation data that are described by text data are rendered on this "png" map in SMS. In order to show the disaster affected areas (points) on the map, Google maps are used to get the degrees of latitude and longitude of two different points. "By calculating the distance between user's location and the point obtained from Google maps, users' location is identified from the reduced scale distance that is started from two different points" [].

1.4 Solution Approach

In this project, we simulated our crisis mapping system using two grids. Both grids represented the same area (any area) divided into equally sized squared cells. In the first grid, we let fire and mobile phone sensors spread freely over the cells. In the second grid, we estimated the fire situation based on mobile sensor readings from the first grid using Gaussian Processes Regression (GPR). Having the actual situation put next to the estimated situation, we could easily verify the strength and weaknesses of both the crisis mapping system and concept in terms of similarity and differences.

1.5 Report Outline

Chapter one introduces crisis mapping, what problem this project is trying to solve and the solution approach. In addition the chapter contains literature reviews of relevant papers. The second chapter of the report contains the theoretical background which the project is based on. The fire theory is used in the implementation of the fire simulation and the theory on Gaussian Processes is used in the implementation of the interpreter.

Chapter tree describes the project requirements and limitations. Additionally it outlines what the simulation and interpreter were expected to do, how the expected functions were implemented and if our way of implementing the features were the correct one. The fourth chapter contains the discussion which aims to present the results, explain why the solution approach was chosen over other alternative solutions and how well the solution satisfied the requirements.

Finally, chapter five contains the conclusion. It endeavours to determine whether the results makes a difference to existing solutions and what the benefits are for system users. Additionally it contains proposed next steps if the project were to be continued.

Chapter 2

Theory

2.1 Gaussian Processes for Regression

$$k(x, x^{'}) = \sigma_f^2 exp\left[\frac{-(x - x^{'})^2}{2l^2}\right] + \sigma_n^2 \delta(x, x^{'}) \tag{2.1}$$

$$K = \begin{bmatrix} k(x_1x_1) & k(x_1x_2 & k(x_1x_3 & k(x_1x_4 & k(x_1x_5 \\ k(x_2x_1) & k(x_2x_2 & k(x_2x_3 & k(x_2x_4 & k(x_2x_5 \\ k(x_3x_1) & k(x_3x_2 & k(x_3x_3 & k(x_3x_4 & k(x_3x_5 \\ k(x_4x_1) & k(x_4x_2 & k(x_4x_3 & k(x_4x_4 & k(x_4x_5 \\ k(x_5x_1) & k(x_5x_2 & k(x_5x_3 & k(x_5x_4 & k(x_5x_5 \end{bmatrix} \tag{2.2}$$

$$\begin{aligned} x - x^{'} &= Distancebetweenobservationdata \\ l &= lengthparameter \\ \sigma_f^2 &= Maximumallowablecovariance \end{aligned} \tag{2.3}$$

Chapter 3

Solution

3.1 Fire and Sensor Simulation

3.2 Fire Interpreter

The fire interpreter's job is to receive input from the simulator and calculate values for the received cells. The calculated values are then posted to the visualizer. The advanced regression calculations are done by the library pyXGPR. This is a Gaussian Process Regression library implemented with Python. It produces a mean and a variance when used correctly. The first input parameter **X** is a list of points which tells where the training data is located. Another parameter **Y** contains the values to the training data. The last interpreter generated parameter **x star** contains the points where we want to find the mean and the variance. In addition to these parameters the library needs to be told what covariance functions pyXGPR should use to calculate the correlation between the cells in **X**, **Y** and **x star**. There is also added parameter values to these functions.

The most basic use of pyXGPR is one dimensional (line regression) where **X** is the location and **Y** is the value. The fire interpreter uses regression in three dimensions where **x** and **y** are the map coordinates and an additional parameter **t** is for time. **t** is necessary to save earlier sensor data which later are utilized in predictions. It should also be mentioned that before this implementation, this was done by saving the best data. Best data is to be understood as the data which has the lowest variance. Data with lower variance would be applied to the saved map. This hack and the implementation of t is done because previous sensor data is important as long as they are weighted less than the newest sensor data. As time

increases there will be sensor data covering a larger portion of the map, but the old sensor data will have less weight and thus giving new sensor data the opportunity to be taken into account. Figure 3.1 illustrates this.

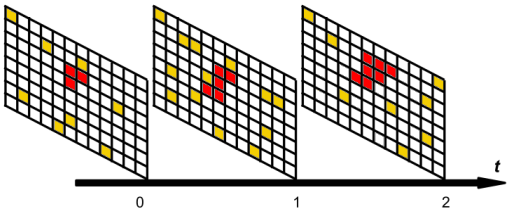

Figure 3.1: The fire interpreter saves the sensor data (yellow dots) and gives them a timestamp **t**. Earlier data is weighted less since the current **x star** and **X** has the same and last value for **t**.

3.2.1 Performance

The simulation map is 71 X 71 squares. GPR calculates the correlation between all points. This means pyXGPR creates huge matrices. As **t** increases the matrices grow larger and thus the execution time rises. The first version of the program used more than 6 minutes to run on a normal laptop. To better the performance it was found necessary to locate some squares which did not need to be calculated. Cells containing sensors was therefore removed from **x star** since the values were already there. With the introduction of **t**, the matrices became larger and execution time rose and once again it was necessary to find some steps to improve the performance. For a cell to be added to **x star** it had to be close to sensor which communicated fire. This was done by finding the euclidean distance between all cells which had no value and all sensors which communicated fire. This reduced the number of points in **x star** significantly. As time increases Another measure taken to reduce the execution time was to save cells which was calculated to be on fire. From these the euclidean distance to all points were calculated and if they were within a certain distance they would be added to **x star**.

Before the interpreter posts the calculated data to the visualizer it converts the mean values to discrete values which is used in the visualizer to decide how intence the fire is burning. A threshold to determine if there should be fire is set, but can be difficult to determine as time progresses. The input values for sensors sensing fire is 0 to 10, while sensors which are not sensing fire is converted from 0 to -1. The reason for the conversion is to get some more distance between fire

and not fire. The fire threshold is set low to make sure all the realistic fire is covered.

3.2.2 Changes in pyXGPR kernel to support wind

The simulation use wind as a crucial parameter to decide where the fire is spreading. Wind is therefore clever to use in the interpretation process. To make it a useful parameter the kernel in pyXGPR has been edited. The kernel contains all the different covariance functions and noise functions. The modification has been done in the function which measures the squared distance between two points. The squared distance is measured by creating two matrices for each dimension. In the fire interpreter these dimensions are **x**, **y** and **t**. One matrix containing all sensor data is deducted from a matrix containing all the uncalculated data. The dimensional result is multiplied with itself and added to a distance matrix. All dimensional results are added to the distance matrix.

$$\theta = cos^{-1}\left(\frac{vector_{ij} \times windVector}{\|vector_{ij}\| \times \|windVector\|}\right) \tag{3.1}$$

$$weight = \frac{\theta}{\pi} \tag{3.2}$$

$$\begin{bmatrix} distance_{11} & distance_{1j} \\ distance_{i1} & distance_{ij} \end{bmatrix} \times \begin{bmatrix} weight_{11} & weight_{1j} \\ weight_{i1} & weight_{ij} \end{bmatrix} \tag{3.3}$$

Before these calculations the interpreter implementations in this function creates a weight matrix which has the same column and row values as the distance matrix. This weight is calculated with the forumula in 3.2. The Hadamard[2] product (see equation 3.3) of these two matrices is returned as the new distance matrix.

Where $vector_{ij}$ is the vector from sensor position i to uncalculated position j. The weight is further normalized to better fit the wind. If $vector_{ij}$ is between $90°$ and $180°$ the weight will be normalized to larger than one and less than one if the angle is less than $45°$. The values in the weight matrix is multiplied with the values in the distance matrix to create a hadamard[2] product. $vector_{ij}$ in the weight

matrix is multiplied with $vector_{ij}$ in the distance matrix. This modifies the distance matrix where some values are reduced and some are increased, determined by their vector direction. The correlation of cells looks at the distance between them. Therefore decreased distance to a sensor sensing fire gives a cell a higher probability of being on fire. Sensors which does not detect fire will have a weight of one.

3.2.3 Unexpected predictions with wind

Figure 3.2: The left image illustrates when wind is used in predicting fire, while the right image illustrates the faults with a pure wind implementation. The blue dots are where the there should have been predicted fire.

The implementation of wind is working, but has some drawbacks. The first image in figure 3.2 illustrates how the fire is predicted when the wind is blowing from the east. The spread starts where the sensor is located and continuous in the shape of a triangle towards west. This looks good in the first image, but the second image illustrates the drawbacks. The wind is still blowing from the east and the two sensors detecting fire are on a horizontal line. The right tips of the predicted fire is where the sensors are located. The blue squares is used to highlight where there should have been predicted fire. This situation occurs because the blue squares are closest to the second sensor. The vector which goes from the second sensor to the any of blue square has an angle which is more than $45°$ when compared to the wind vector which is $[-1, 0]$. Problems with predicting the middle part of the fire can occur when the the size of the predicted fire is becoming quite large. To solve both these problems there has been developed two solutions which have been tested with varying results. The first solution looks at the correlation between all sensors sensing fire while the second solution uses some techniques found in graphical programming. Another approach to these problems would be to use dynamic parameters. As the fire progressed the parameters would change in accordance to the size of the fire. But problems arises with such a solution as well. The edge of the fire would be more likely to be smudged out and it would in some cases predict fire in areas where there were no fire.

Solution 1 to unexpected predictions with wind

Burning sensor correlation was one approach to solve the problem when applying wind. It creates a list of all sensors which are sensing fire. Each element in the list has vectors to all other sensors communicating fire. In figure 3.3 the green triangle is an uncalculated point while the numbered red circles are sensors sensing fire. There are vectors from sensor 1 to all other sensors sensing fire and a vector to the green triangle. This is an illustration of one of the entries in this list. The basis for this theory is that there is probably actual fire between the sensors. The blue vector compares direction with all the black vectors, see 3.1. It finds the black vector which has the most similar direction and check if it is within a certain threshold. If this comes out positive the distance for this point to sensor one will be multiplied with a number less than one, else it would be multiplied with one. This approach

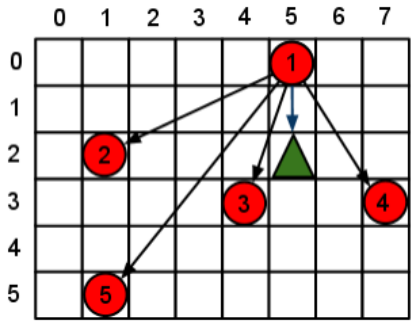

Figure 3.3: All sensors (red) are sensing fire and sensor one has a vector to all other vectors sensing fire. The green triangle is an uncalculated point. Burning sensor correlation looks for the black vector which has the most similar angle to the closest blue vector.

has a fault. When a situation like 3.2 (second image) occurs it would predict fire against the wind. The burning sensor correlation would neutralize the added wind to such a degree that it was removed. The complexity with this solution is high and thus the execution time rises fast as **t** increases and the fire interpreter uses more data. As this solution was developed it would handle multiple fires badly.

Solution 2 to unexpected predictions with wind

In the second approach to solve the wind and filling issues, components from computer graphics were used. The outer boundary of the sensors sensing fire was chosen. The shape will be a convex polygon. Uncalculated cells within this polygon will have a higher probability of being on fire in the prediction. As in the

burning sensor correlation implementation it is assumed its a higher probability of being fire between sensors sensing fire. To locate all the outer cells Graham's

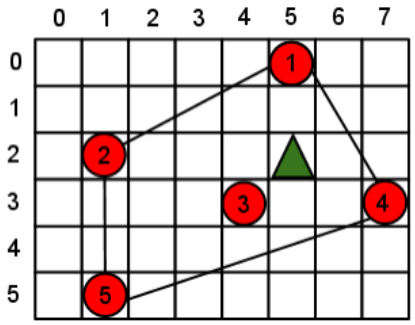

Figure 3.4: Locates all cells within the boundary.

scan [1] was applied. When these points were found Bresenham's[3] line algorithm was used to find the in between cells. The result was all cells which the lines covered in figure 3.4 was retrieved. After this a scan line fill algorithm was used to find all the cells within the polygon. These steps resulted in a list of cell positions. This list was used when calculating the wind in accordance to wind direction. If the cell to be evaluated was inside this polygon the weight was multiplied with a number lower than one. This solution is also much faster than the first one. The Graham's scan and Bresenham's line algorithm was used as third party code, meaning we did not implement them. But the scan line algorithm was implemented in the kernel. Currently it does not handle multiple fires. If the two fires started at different **t** it could be possible to walk on predicted fire from one sensor to all the others.

Chapter 4

Discussion

4.1 Results

The results are divided into two groups, simple simulation and advanced simulation. The results addressed in this section are reasonably representative for the average results.

4.1.1 Results with simple simulation

The simple simulation is characterized by sensors being spread relatively even throughout the map. The rule is that a sensor cannot touch another sensor. When using the simple simulator each sensor can only sense its closest neighbours.

Figure 4.1 illustrates the success of the fire interpreter. The light green color is where the actual fire is and where the interpreter predicted it to be. Therefore the color of success. The dark green color is area where the interpreter thought it would be fire, but was not. The red parts are the actual fire which the interpreter did not predict.

4.1.2 Results with advanced simulation

The advanced simulation mimics the effect of humidity and wind has on a fire. The sensors are spread more randomly than in the simple simulation. They can also sense with a larger range. The default is two cells. The fire interpreter used the wind add-on in the kernel in the successful tests, while it was turned off in

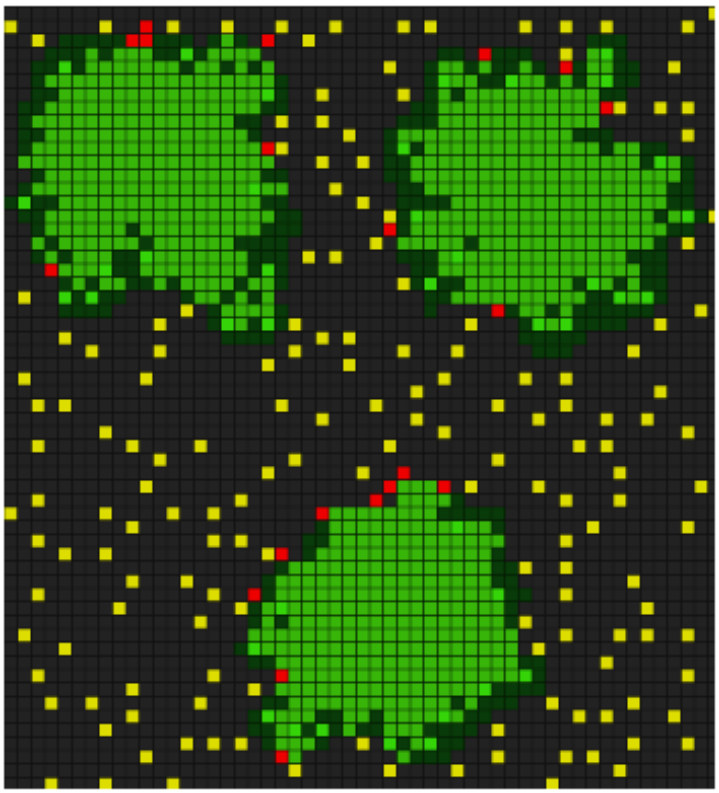

Figure 4.1: The intense green is where the fire interpreter predicted correctly. The dark green is where the fire interpreter thought it was fire, but it was not. Red dots are where there actually was fire, but the predictor was unable to predict it.

the tests which did not have too good results. There are a number of parameters which has been tweaked to get good results. These different parameters makes it more difficult to get an accurate prediction. Figure 4.2 illustrates the predicted fire on top of the actual fire when the wind is blowing from west. The prediction is covering a larger area than the actual fire. This is because the sensors are sensing 2 cells away. In figure 4.2 this can be observed with the right most predicted fire. The two sensors which lays beneath the right most predicted area is sensing fire and therefore the prediction is set from this point. The first cells north and south of this area has also been calculated to be on fire. This is because of Bresenham's line algorithm which interprets these two cells to be on the outer boundary of the sensors sensing fire. The algorithms from computer graphics is used to make sure there are no holes inside the predicted fire and to prevent wind problem (chapter 3.2.3).

Figure 4.3 illustrates what is happening when wind and the graphical algorithms

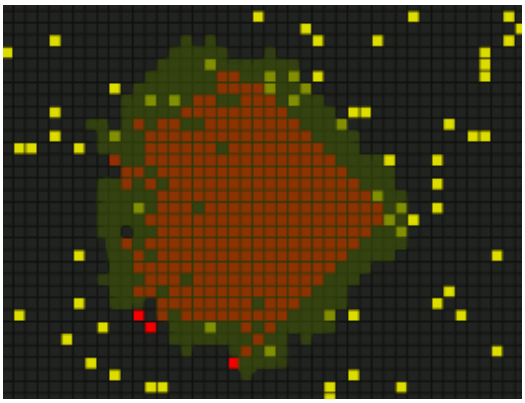

Figure 4.2: The dark green overlay is where the fire interpreter predicted the fire to be. The red covered by the green overlay is where the fire interpreter predicted correctly and the red dots without any overlay is where there was fire, but did not predict.

are disabled. Within the domain of the advanced simulation the results achieved with wind and the graphical algorithms. The prediction with this set up reproduces the shape of the actual fire with a satisfactory result. The area of the shape is larger than the actual fire. The philosophy was always to air on the side of safety. Meaning that the fire interpreter aimed to detect all fires and as such a few false positives were deemed to be acceptable.

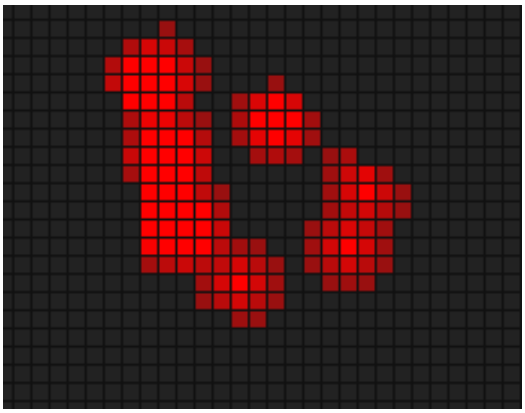

Figure 4.3: Wind and graphical algorithms disabled in the advanced simulation creates holes and shows a lack wind direction.

Chapter 5

Conclusion

The goal of this project was to help the users determine the best course of action during an evolving crisis. As crisis mapping includes a fairly broad range of crisis situations the goal was to create a prototype crisis mapping system that could detect fire using mobile phone sensors. This system were to exclusively use simulated data as setting up mobile phone sensors to properly detect a fire was outside of the project scope. To reach the project goals the system would utilize two grids side by side, one containing the simulated crisis and the other would visualize the data predicted by the Gaussian Processes.

In the end the project met the requirements. In a simple environment the Gaussian Processes have fairly good results. Nevertheless the Gaussian Processes does not return a perfect prediction. Only a few grid squares that are on fire will go undetected, however a somewhat larger amount of false positives will occur. When more environmental factors are introduced to the system the results are less accurate, however they are still within a reasonable range.

While the system itself is not ready for deployment it provides insight into the problem and a starting point for future work. The project was never meant to become a finished product. The system is a proof-of-concept that demonstrates the validity of using Gaussian Processes to predict fire spread. The system provides an easy way of estimating results and as such provides a good framework for future work. There are a few improvements that could be made to advance the current solution.

Firstly the simulation would need to implement a large variety of different squares. In the real world fire is hard to predict as it is affected by several factors. To properly simulate a real fire the simulation would need to simulate a real geographical area with houses, ocean, rivers and so on. Secondly fire would need

to behave according to the current understanding of fire and fire movement. Fire moves faster uphill than downhill, there are different types and shapes of fire and so on.

Thirdly the simulated sensors move randomly in the current version of the system. To improve upon this it could be beneficial to look into human movement during a crisis and have the sensors move accordingly. Finally, if possible the system should receive and interpret real data from real sensors. This would naturally be the best solution, however the system would likely need to go through several versions of the simulator before it could be used to interpret vast amounts of real data.

Acknowledgments

We would like to thank our supervisors Ole-Christoffer Granmo for his constructive feedback that has led to progress in times when the project was at a stand still.

University of Agder, 2011

Bibliography

[1] Alejo Hausner, CS Department, Princeton University *http://www.cs.princeton.edu/courses/archive/spr10/cos226/demo/ah/GrahamScan.html (2011).* Prentice Hall

[2] Teknomo K. Hadamard Product. 2011. Available from `http://people.revoledu.com/kardi/tutorial/LinearAlgebra/HadamardProduct.html`

[3] Flanagan C. The Bresenham Line-Drawing Algorithm. 2011. Available from `http://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html`

http://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html