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

Faculty of Engineering and Science

Department of information- and communication technology

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

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

Theoretical Background

2.1 Gaussian Processes for Regression

Mathematical solutions such as linear regression are often used to predict unknown values based on similar values. Depending on input data, output accuracy and simplicity, mathematicians are encouraged to use several other methods as well. Gaussian Process Regression (GPR) is one of the popular methods has been used due to several reasons such as:

- Very accurate output
- Less number of variables
- Learning by experience

Correct predictions must be based on the correct assumptions. Even though GPR is learning from experience, it is not "free form" which means it is not automatically generated[ ]. Other techniques, such as "squared exponential", can be used when the users are unable to do even basic assumptions.

It is acceptable to assume that each observation point has data distribution similar to normal distribution. Therefore, it is needed to calculate the mean value and variance for all predictions.

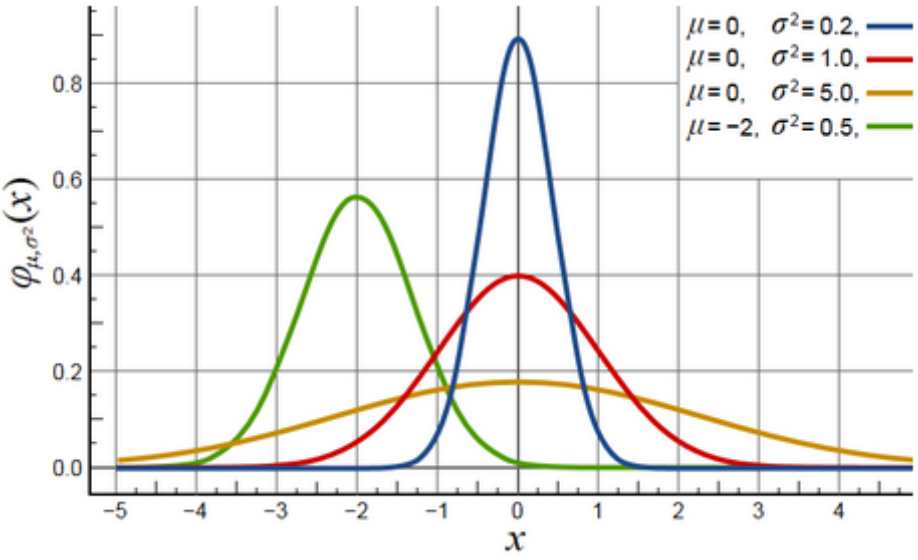

Figure 2.1: Normal Distribution curves[4].

2.1.1 Covariance Function

In general, users can assume that "partner GP's mean is zero" everywhere[6]. Therefore, two similar cases are related to each other by a covariance function $k(x-x')$. Among the various options "squared exponential" is one of the popular choices for the covariance function.

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

$$\begin{aligned} x - x^{'} &= Distance\ \ between\ \ observation\ \ data \\ l &= length\ \ parameter \\ \sigma_f^2 &= Maximum\ \ allowable\ \ covariance \end{aligned} \tag{2.2}$$

To predict accurate data for a smooth curve, input (or observed) data from the neighbors must be identical. By studying the above equation, we can understand that σ_f^2 is increased for functions which cover a wide area through the y-axis. When the $(x-x')$value becomes larger, $k(x-x')$ approaches zero depending on the length of parameter l.

The effects of the length parameter l, can be explained as follows: Lets assume $l = 0.1$ for a particular prediction and graph $f(x)VSx$, which is shown in figure 2.2. It is clear that the curves are not aligned[7]. So, we change $l = 0.3$ and

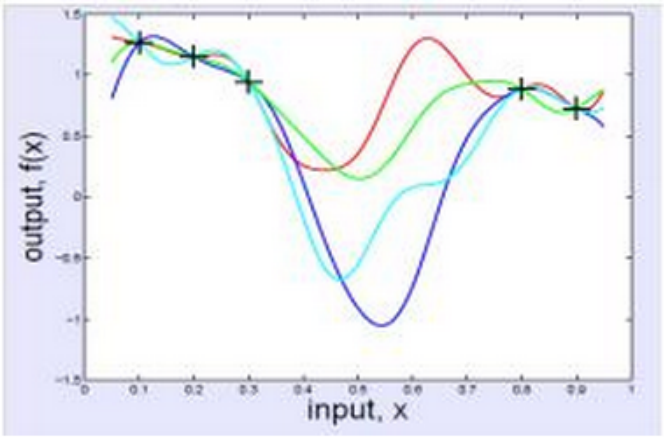

Figure 2.2: When $l = 0.1$. [5].

evaluate the results. This is shown in figure 2.3. It is clear that the curves are more aligned here than in figure 2.2. Since the curves are not perfectly aligned,

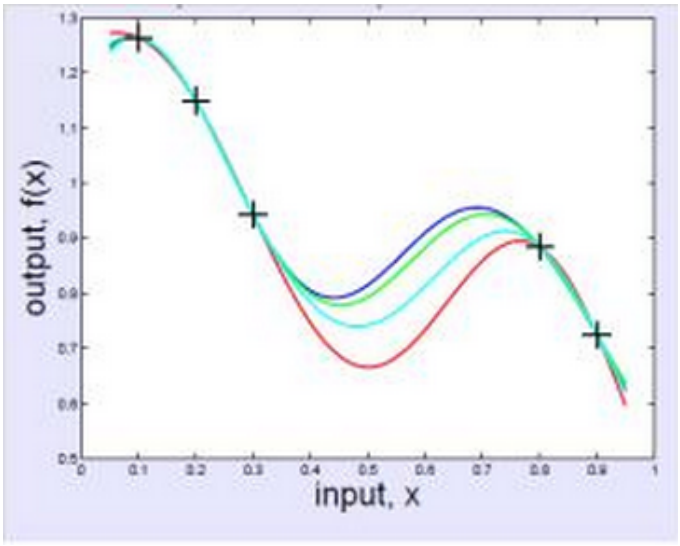

Figure 2.3: When $l = 0.3$. [5].

length parameter l, is increased further and result is evaluated. In figure 2.4 we set $l = 0.5$ and evaluate the results. Here we can observe that the curves are perfectly aligned [5]. Apart from this, function consists of additional part to represent errors. This is an important part, when GPR is used to predict practical events. (e.g.

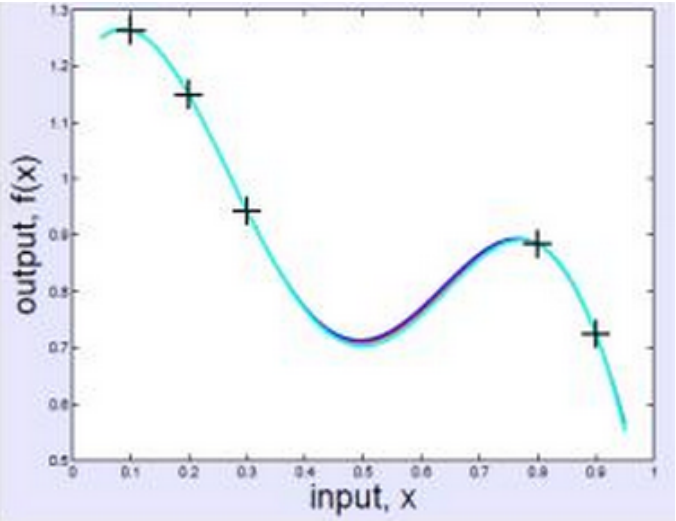

Figure 2.4: When $l = 0.5$. [5].

environment temperature can be vary, due to external factors such as wind).

$$k(x, x^{'}) = \sigma_f^2 exp\left[\frac{-(x - x^{'})^2}{2l^2}\right] + \sigma_n^2 \delta(x, x^{'}) \tag{2.3}$$

However, people preferred to keep σ_n^2 separately and work on covariance and error to simplify the calculation.

2.1.2 Method of Calculation

The covariance value, $k(x, x')$ can be calculated with each observed data in respect to each other observations. Values can be represented by a metric. By assuming that five observations have been conducted, $x1, x2, x3, x4, x5$. $k(x, x')$ value for those observations can be written as follows:

$$K = \begin{bmatrix} k(x_1x_1) & k(x_1x_2 & k(x_1x_3 & k(x_1x_4 & k(x_1x_5 \\ k(x_2x_1) & k(x_2x_2 & k(x_2x_3 & k(x_2x_4 & k(x_2x_5 \\ k(x_3x_1) & k(x_3x_2 & k(x_3x_3 & k(x_3x_4 & k(x_3x_5 \\ k(x_4x_1) & k(x_4x_2 & k(x_4x_3 & k(x_4x_4 & k(x_4x_5 \\ k(x_5x_1) & k(x_5x_2 & k(x_5x_3 & k(x_5x_4 & k(x_5x_5 \end{bmatrix} \tag{2.4}$$

This is a symmetric metric, $k(x1, x2)$ and $k(x2, x1)$ where both elements have the same values. All diagonal elements are equal and show the highest $k(x, x')$ value since $\frac{-(x - x^{'})^2}{2l^2}$ becomes zero.

When the position is moving away from the diagonal line of the matrix, the value of $k(x_n, x_m)$ goes towards zero. In other words, if the matrix contain higher number of rows and columns, the values of positions such as far away from the diagonal are almost equal to zero, while diagonal values shows maximum covariance value.

While the K matrix gives the covariance values for observed data points the $K*$ matrix gives the covariance values for considered points compared to other observed points. Let's make the matrix K_* relative to the above five positions.

$$K_* = [k(x_*, x_1) \quad k(x_*, x_2) \quad k(x_*, x_3) \quad k(x_*, x_4) \quad k(x_*, x_5)] \tag{2.5}$$

$K_{**} = k(x_* x_*)$ also can be found same way as it is directly given by $k(x, x') = \sigma_f^2$. From the above values, mean and variance values for a given y_* can be found in the following way:

$$\begin{aligned} \overline{y_*} &= K_* K^{-1} y \\ var(y_*) &= K_{**} K_* K^{-1} K_*^T \end{aligned} \tag{2.6}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} \tag{2.7}$$

When finding the variance, a suitable confidence level must be selected according to the prediction (e.g. 95% confidence level).

2.1.3 Prediction of Temperature Using Gaussian Process Regression

The objective is to predict temperature at a particular point by interpreting other observed data. Imagine the following situation: The area highlighted in gray colour has sensors (Thermometer) to observe the temperature and one would want to predict the temperature on red colour(3,3) in figure 2.5. Since we can find the distance between each cells, it is possible to define a covariance function using observed data (given from thermometers). σ_f^2 and l (length parameter) can be determined by using the observed data. Thereafter, K_{**}, K_*, K matrices can be determined. This will give the values for mean and variance. Finally, it is possible to plot the graph and this will give the predicted values for any positions within the considered area.

1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

Figure 2.5: Value matrix.

2.2 Forest Fire Theory

2.2.1 The Fundamentals of Fire

For there to be a fire, the fire triangle(Showed in figure 2.6) needs to be satisfied. The three components in the fire triangle is an oxidizing agent, fuel and heat. All these tree depend on each other and without one of them, there will be no fire. An oxidizing agent is a substance that removes an electron from another substance. Oxygen is the normal example of an oxidizing agent.

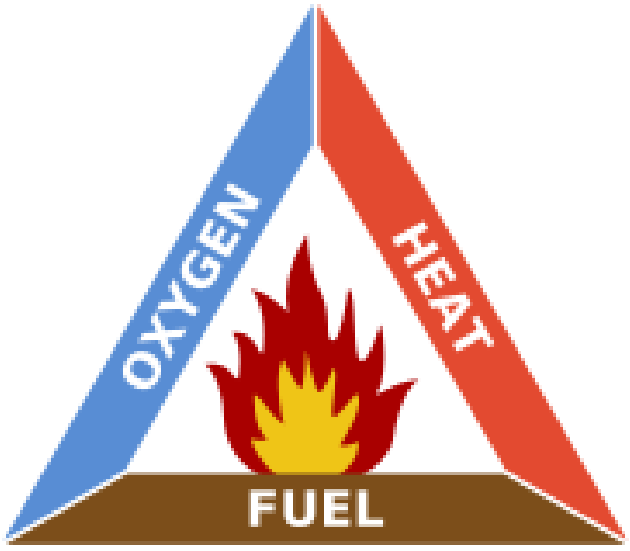

Figure 2.6: Fire triangle.

2.2.2 Analyzing the Shape of Fire

By looking at the shape of a fire, you can determine what part of the fire is a heading fire or a backing fire. Shown in figure 2.7, the heading fire is much lager then the backing fire, this is due to the fact that heading fire moves more quickly. The reason for this is that a heading fire is always going the same way as the wind, and a backing fire is going in the opposite direction of the wind.

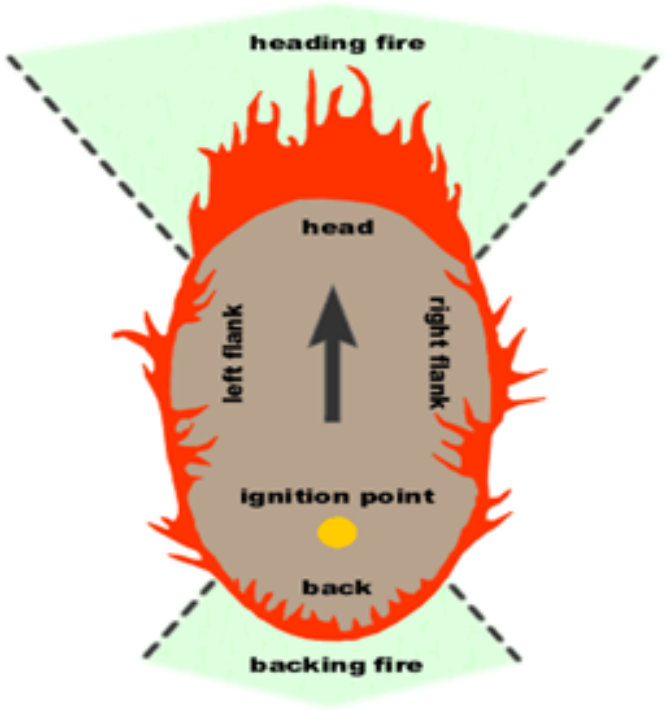

Figure 2.7: This illustration shows different parts of a fire which is ignited from one point.

2.2.3 Fire Movement

The flames tend to move on the uphill side quickly due to the easy access to fuel. Moreover hot convective air from fires moves up the slope drying out fuels which means it will ignite easier as seen in figure 2.8. As a result the fire spreads twice as fast while moving uphill. It is an important fact that fire behavior in complex landscape is unpredictable due to narrow valleys and ridges. As it can suddenly change direction and magnitude. []

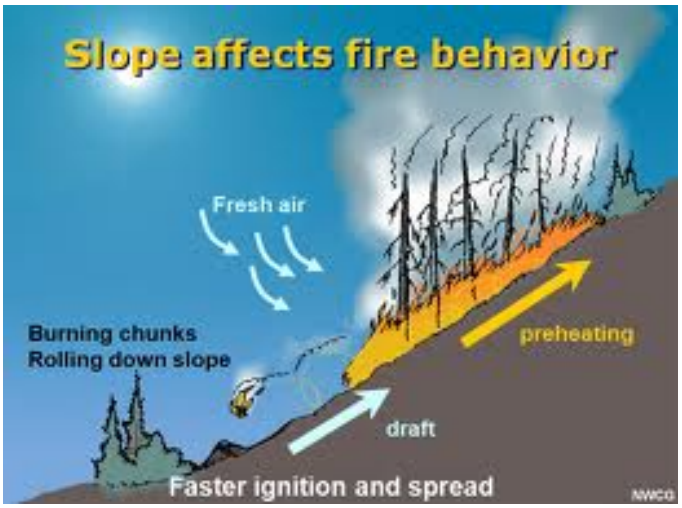

Figure 2.8: Slope affects fire behavior. []

2.2.4 Fire Types

There are different types of fires, surface fire, crown fire and ground fire are tree types of forest fires. The surface, as seen to the left in figure 2.9, fire burns on the forest floor and often use shrubs, grass, broken branches, litter and leafs as the source of fuel. When they grow large enough, almost anything will do as fuel. Crown fires, as seen to the right in figure 2.9, are always ignited by surface fire, this is the fire in trees or bush tops. Fires on the ground may use subsurface organic material as fuel, these fires may be ignited from surface fire. Soil with a high organic percentage such as peat found in the arctic tundra, bogs and swamps tends to be the fuels to these kinds of fires.

Figure 2.9: Surface fire and crown fire. []

2.2.5 Wind

Wind is one of the most important factors in controlling and changing rate and direction of the spread of fire. Figure 2.10 shows how the wind changes the direction of fire. As a rule the fire changes direction wherever the wind blows. First the wind direction is from north west, which means the fire spreads south east. After the wind has changed direction the fire follows and now spreads towards north east.

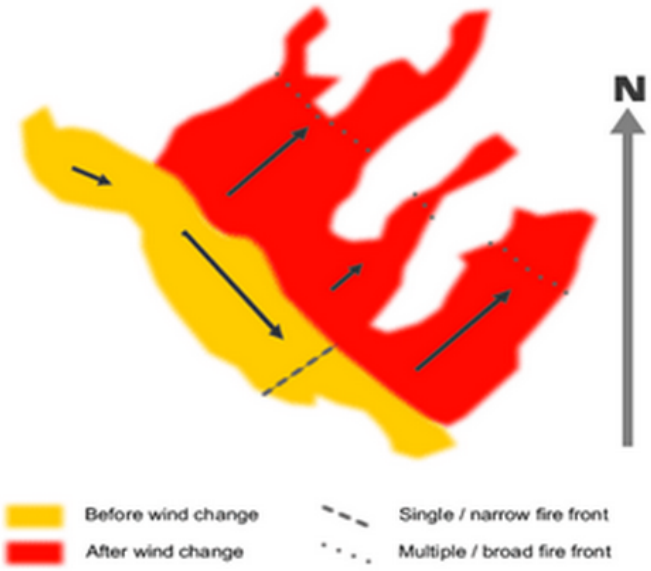

Figure 2.10: Wind affects fire. []

2.2.6 Climate and Vegetation Effect on Fire

As illustrated in figure 2.11 climate has a strong influence on vegetation and fire. This is illustrated by the thickness of the arrows . For instance climate has a far-reaching effect on vegetation type, while vegetation and fire has a small effect on the climate. The climate has a big impact on the vegetation. For example a tropical rain forest is a result of high humidity and high temperatures. If we remove the high humidity savannah and grassland will appear. Fire will occur when the weather is hot, dry, windy, lightning or low humidity. With all these conditions in place in addition of a temperature rise the chances of an intense fire is present. On the other hand fire would rarely occur if precipitation and the moisture in the fuel is high. Moreover fire has a huge influence on vegetation type,

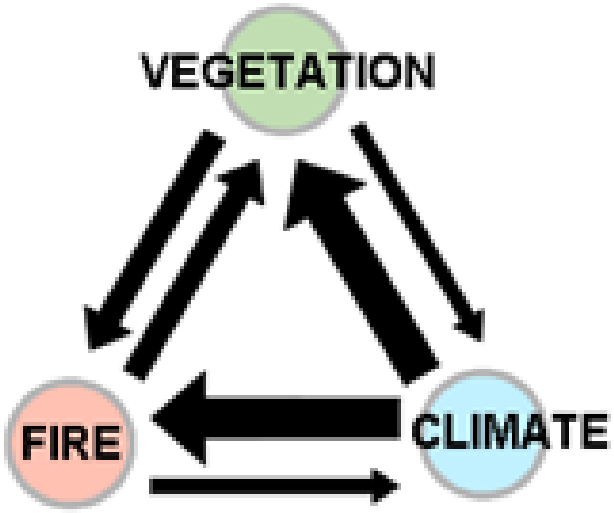

Figure 2.11: Climate, vegetation and fire create a chain as follows: []

it can be damaging to the vegetation in rain forests because it has problems with the extreme changes. Similarly vegetation can affect fire. Some plants has a high level of moisture while others can be dry. This difference in flammable degree can have a noticeable impact on how the fire spreads and makes it harder to predict how the spread will be.

2.2.7 Temperature and Flame Height

The temperature of a forest fire depends on some conditions like climate and type of fuel. An average forest fire have a temperature up to $800°C$, and a flame height up to 1 meter. In some extreme cases the temperature can exceed $1200°C$ and flame heights can become over 50 meters. The higher flames are, the more crown fires will occur during a forest fire.

2.2.8 Relative Humidity

Relative humidity have a huge impact on water content in small fuel sources as water vapor can easily penetrate into or escape from the center of small fuel sources. Thus wood with a high amount of water content will ignite more slowly compared to dry wood. The reason for this is that the water inside of the wood

needs to vaporize before the wood can burn, and the vaporization of this water may require a lot of energy. In addition, high water content will cause a fire to burn with a lower intensity as the water has to vaporize before the fire can reach its maximum intensity [].

2.2.9 Rate of Spread

The time it takes for a fire to move over a horizontal distance is the Rate of Spread. Since the rate of fire may vary a lot from any point the fire is spreading, it is usually an average value. The head fire will have the highest rate of spread, the back fire will have the lowest and the flanking fire will have a medium rate of spread. An easy way to measure the rate of fire is to find two or more landmarks of a known distance from each other, then take the time for the fire to pass the landmarks. There are also other techniques that can be used to measure the rate of spread, like using a video camera or placing firecrackers [].

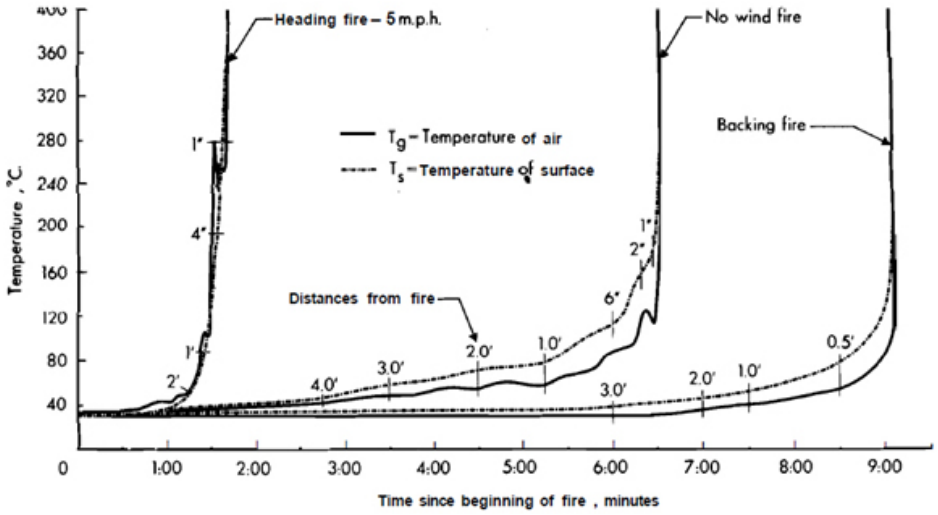

Figure 2.12: Mathematical Model for predicting fire spread in forest fire. []

Formula to calculate rate of spread

Rothermel's equation have been the basis for most of the fire spread prediction models. One formula to calculate the rate of spread was mathematically expressed by this equation. This equation as shown in figure 2.12, is the heat received by the

fuels ahead of the fire divided by the heat required to ignite the fuels.

$$R = \frac{IR\xi(1 + \Phi W + \Phi S)}{\rho\eta\varepsilon Qig} \tag{2.8}$$

- R = rate of spread of the flaming front
- IR = reaction intensity
- ξ = proportion of the reaction intensity that heats adjacent fuel particles to ignition
- ΦW = dimensionless multiplier accounting for the effect of wind in increasing the proportion of heat that reaches adjacent fuels
- ΦS = dimensionless multiplier accounting for the effect of slope in increasing the proportion of heat that reaches adjacent fuels
- $\rho\eta$ = ovendry fuel per cubic foot of fuel bed (lb/ft)
- ε = dimensionless number accounting for the proportion of a fuel particle that is heated to ignition temperature at the time flaming combustion starts (near unity for fine fuels and decreases toward zero as fuel size increases)
- Qig = heat of pre-ignition, or the amount of heat required to ignite one pound of fuel (Btu/lb)

This formula can be used to properly describe rate of spread. This could later be used in combination with other factors, such as wind direction and terrain to correctly implement fire behaviour.

Chapter 3

Solution

3.1 Implementation

3.1.1 Grids

In the implementation, we introduced a concept called grid. A grid was a visual representation of an area (any area) divided into discrete valued cells. The purpose of the grid was to easily view the actual situation and the predicted situation after reconstructing it based on samples from the actual situation (mobile phone sensor readings). As figure 3.1 shows, we used two separate grids placed next to each other where the left grid is always the actual situation and the right grid is always the predicted situation. These two grids served as a guide line throughout the implementation. If the grids did not look somewhat similar at all times, some error had been done. How these grids were used are explained in the following sections.

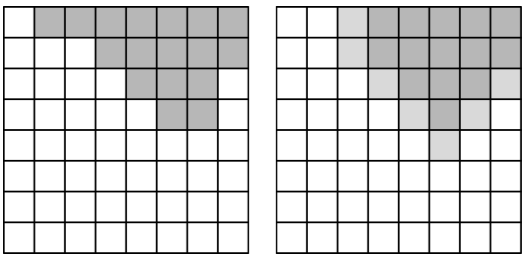

Figure 3.1: Two grids are used to compare actual situation on the left and the estimated situation on the right.

3.1.2 Simulating Data

Simulating data was done in the left grid. At launch, the user had to choose a location for the initial fire. A user could start several fires at once, where each one would spread independent of the other fires. The spreading was done automatically for each time iteration.

The fire spread was implemented with a simple algorithm. Given a two-dimensional grid of cells, the more burning neighbours each cell C on the grid had, the higher the chances were for it to ignite. Because there is a maximum of eight neighbors, we divided the number of burning neighbors by the total number of neighbors as shown in 3.2

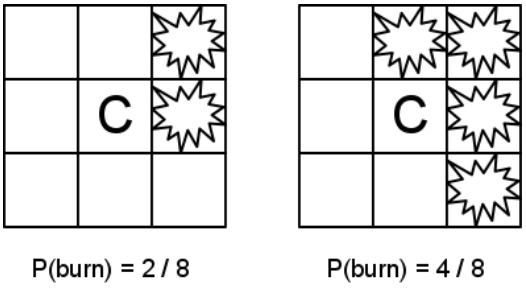

Figure 3.2: Probability that fire will spread to a cell C on the grid without fire.

The simple probability calculation was later extended to account for wind. Since wind can blow away smoke and heat wind direction and magnitude directly affects how much impact each neighboring cell would have on C. To simulate this, each neighboring cell used a portion of the wind vector to produce a weight of impact. The sum of all these weights divided by a total weight became the new probability calculation in the fire spreading algorithm.

Humidity was implemented after wind. Humidity affects how fast fire spreads. It was implemented as a global value and had the same value for each cell. With this simplicity, we only had to either increase or decrease the total weight used in the probability calculation to adjust the rate of fire spread.

3.1.3 Sensing Data

After simulating data with wind and humidity, the next step was to use mobile phone sensors to read this data. Even if the current situation was known through the simulation at all times, we were only supposed to know the situation by the mobile phone sensor readings. To simulate sensing, two additional algorithms were used.

The first algorithm was a walking algorithm to simulate that people walk around with the mobile phone sensors. This algorithm assumed that sensors already had a position in order to do the following. For each time step, there was a fifty percent chance for each sensor to move. Because there were eight different directions to walk in a grid, the sensor picked a random direction. If the neighboring cell in the picked direction was on fire, the sensor had to pick another direction. In all, this algorithm only handled the location of the mobile phone sensors in the simulation. The next step after placing the sensors was to do readings.

The second algorithm was a sensing algorithm to provide known data used by the crisis mapping system. Because heat and smoke moves in the direction of the wind, the resulting sensor readings highly depended on wind and the location of the fire. Consider the example in figure 3.3. Given the same wind vector W and two different neighbors at relative positions V1 and V2, each neighbor contributes differently to the sensor in C. In this particular case, the wind either blow heat and smoke away from C or into C depending on which neighbor it is.

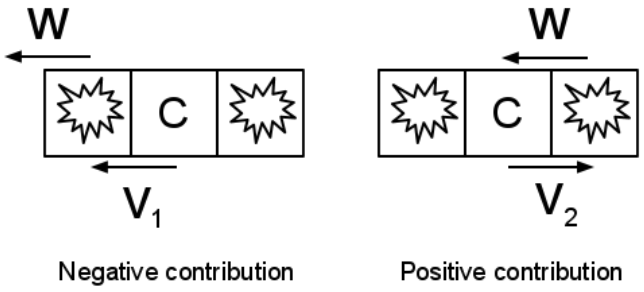

Figure 3.3: Sensor readings are affected by wind and the positions of the neighboring emitting cells.

The heat and smoke contribution to the sensor at C could be calculated as the sum of the cosine value of the angle between the wind vector W and the position

vector for each neighbor. As shown in figure 3.4, the cosine value could be divided into two ranges.

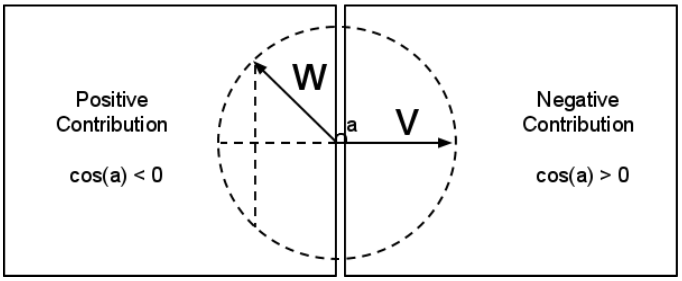

Figure 3.4: Neighbor heat and smoke contribution is found by the cosine value between on wind W and neighbor position V relative to the sensor.

The range [-1,0] was equal to positive contribution while the remaining [0,1] was equal to negative contribution. In case of a negative contribution, the contribution can simply be treated as 0 because heat and smoke would blow away from the sensor. To further simplify, by taking the absolute value of the positive contribution range [-1,0], the new range became [0,1] where 0 is the lowest and 1 is the highest contribution. This could be used as a coefficient to the heat and smoke emitted by this neighbor. Given that there are eight neighbors each indexed by i, the resulting mathematical formula used in the sensing algorithm is as follows:

$$c = \sum_{i=1}^{8} \begin{cases} |V_i \cdot W / (|V_i| \cdot |W|)| & \text{if } V_i \cdot W < 0 \\ 0 & \text{if } V_i \cdot W >= 0 \end{cases}$$

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 3.5 illustrates this.

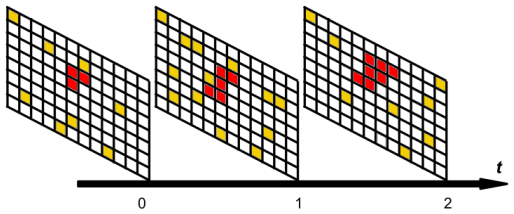

Figure 3.5: The fire interpreter saves the sensor data (yellow dots) and gives them a timestamp **t**. Earlier data is weighted less since the current **x star** and **X** has the same and last value for **t**.

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 3.6: 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 3.6 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 3.7 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

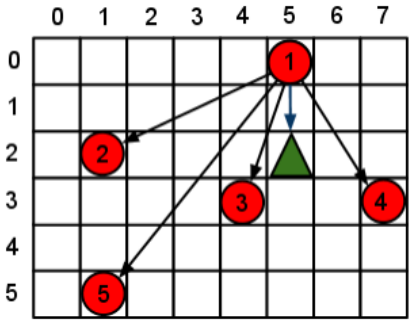

Figure 3.7: All sensors (red) are sensing fire and sensor one has a vector to all other vectors sensing fire. The green triangle is an uncalculated point. Burning sensor correlation looks for the black vector which has the most similar angle to the closest blue vector.

has a fault. When a situation like 3.6 (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

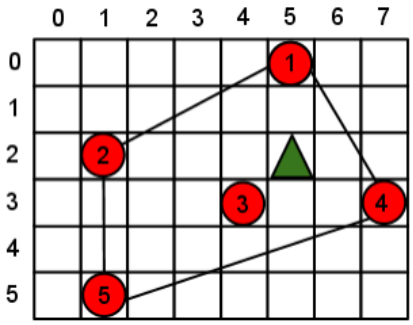

Figure 3.8: Locates all cells within the boundary.

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 3.8 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

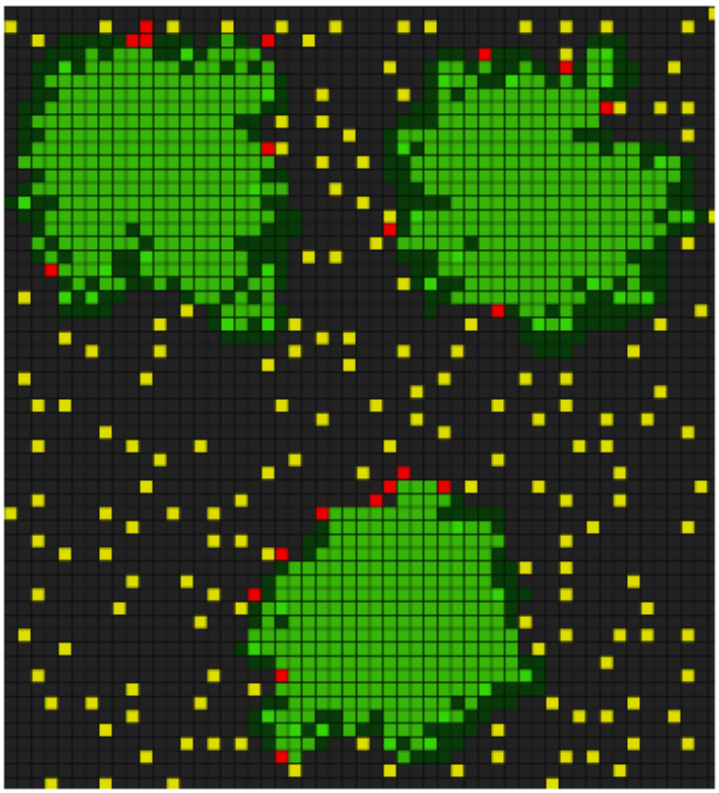

Figure 4.1: The intense green is where the fire interpreter predicted correctly. The dark green is where the fire interpreter thought it was fire, but it was not. Red dots are where there actually was fire, but the predictor was unable to predict it.

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

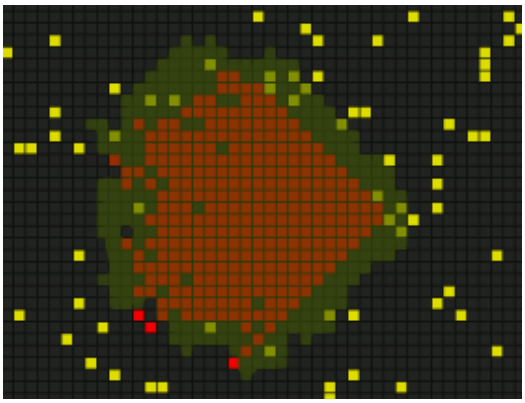

Figure 4.2: The dark green overlay is where the fire interpreter predicted the fire to be. The red covered by the green overlay is where the fire interpreter predicted correctly and the red dots without any overlay is where there was fire, but did not predict.

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.

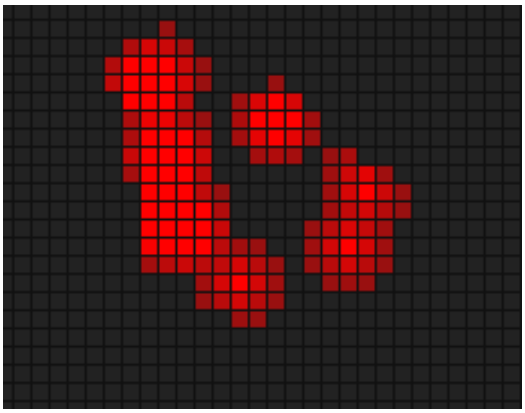

Figure 4.3: Wind and graphical algorithms disabled in the advanced simulation creates holes and shows a lack wind direction.

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`

[4] Inductiveload. Normal Distribution: curve. 2008. Available from `http://en.wikipedia.org/wiki/File:Normal_Distribution_PDF.svg`

[5] Wallach H. M. Introduction to Gaussian Process Regression: curve. 2005. Available from `http://www.cs.umass.edu/~wallach/talks/gp_intro.pdf`

[6] Ebden M. Gaussian Processes for Regression: A Quick Introduction. aug 2008. Available from `http://www.robots.ox.ac.uk/~mebden/reports/GPtutorial.pdf`

[7] Wallach H. M. Introduction to Gaussian Process Regression. jan 25, 2005. Available from `http://www.cs.umass.edu/~wallach/talks/gp_intro.pdf`