SAARLAND UNIVERSITY

Faculty of Natural Sciences and Technology I
Department of Computer Science

Concepts, Implementation and Evaluation of spatio-temporal Visualizations of Bushfires

Bachelor Thesis

André Zenner
Bachelor's Program of Computer Science
September 2013

Supervisor:
Prof. Dr. Antonio Krüger, German Research Center for Artificial Intelligence,
Saarbrücken, Germany

Reviewers
Prof. Dr. Antonio Krüger, German Research Center for Artificial Intelligence,
Saarbrücken, Germany
Dr. Jörg Baus, German Research Center for Artificial Intelligence,
Saarbrücken, Germany

Submitted
16th September, 2013

Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Campus - Building E1.1
66123 Saarbrücken
Germany

Eidesstattliche Erklärung:

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath:

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Saarbrücken, 16th September, 2013

Einverständniserklärung:

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent:

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, 16th September, 2013

Acknowledgements

At this point, I want to express my appreciation to everybody that helped me writing this thesis.

First and foremost, I want to thank Professor Antonio Krüger for offering me the great opportunity to work on this exciting topic, for motivating me during the whole process, for his always kind, patient and competent support and for his time and effort without which this thesis would not have been possible.

I also want to thank Professor Matt Duckham, Doctor Susanne Bleisch, Lisa Cheong and Professor Allison Kealy from the University of Melbourne for the collaboration and their kind and helpful support and feedback from Australia.

In addition, I would like to thank Denise Paradowski for taking her time introducing me to the survey-system and I am thankful to Michael Piotrowski and Frederic Kerber for the technical support regarding LaTeX.

Special thanks goes to my parents for their encouragement and their support during my whole studies, as well as to my friends and fellow students for the helpful ideas that came up in formal and informal discussions.

Last but not least, I thank everybody that participated in the online-study.

Abstract

Smartphone applications and websites that provide public information about bushfires in bushfire prone regions are becoming increasingly popular and important. This thesis focuses on new visualizations of publicly available bushfire data. The main goal is the development of visualization concepts and algorithms, their implementation in a prototype system and their evaluation.
A comparison of currently used public visualizations in apps and websites with expert visualizations used by professionals reveals a gap between the very simple and intuitive but less informative public visualizations and the highly informative but very complex and not readily accessible visualizations in expert systems. To fill this gap, seven new approaches for public visualization are developed, aiming to communicate more relevant information for laypersons to enable better decision-making. For generating these more personalized and informative visualizations, additional data about the user's location, the spatio-temporal development of the fire and information about the streets in the environment are used. For each visualization, the pseudocodes for the corresponding algorithms are depicted.
Additionally, an easily extendible prototype system is developed, implementing all different visualizations for testing and demonstration.
A final online study, using visualizations of scenarios generated with the prototype implementation, investigates the communicated threat and the popularity of the currently used and the new visualizations. The results indicate that the communicated threat of some new approaches is higher than the currently used ones'. Moreover, users prefer visualizations providing pre-processed information about the spatio-temporal development of the bushfire and information related to the users position and environment. Due to this, the four most requested visualizations are new concepts.
Eventually, some recommendations for further work in this area are given.

Contents

Chapter 1
Motivation

Fire is a fascinating natural phenomenon with many different qualities. Mankind has always tried to control fire in order to use it for their purposes and since humans are able to make fire, they developed rapidly and included it in many areas of life. But even today, humans still are not able to control the power of fire completely. In situations of dangerous fire hazards, people rapidly become aware of nature's destructive force and they realize, that this force must not be underestimated.

1.1 Bushfires

One phenomenon belonging to the class of dangerous fire hazards threatening many people's live every year is the *bushfire*. According to the common definition, a bushfire is *"any uncontrolled, non-structural fire burning in a grass, scrub, bush, or forested area"* [1]. Other names for bushfires are *brush fire, wildfire, forest fire, desert fire, grass fire, hill fire, peat fire, vegetation fire* and *veldfire* [2]. They commonly occur in Australia because of the hot and dry climate, but of course they also occur in many regions all around the globe, in which the climate dries out the vegetation and makes it possible fuel for bushfires.
Bushfires are characterized *"in terms of their physical properties, their fuel type, and the effect that weather has on the fire"* [2] and they are generally hard to predict. They typically start at a point of ignition and in the following spread with different velocities in several spread directions. Bushfires generally burn at multiple fire-frontlines that move forward and burn down large areas. The movement and mutation of a bushfire depends on many complex variables like e.g. type of fuel (underlying vegetation), wind speed and direction, temperature, the slope of the underlying terrain and so on. Furthermore, changes in those variables like a change of the wind speed or direction can have dangerous effects on bushfires

causing them to jump forward or to instantly change their spread direction.
As difficult as the prediction of the fire's spread is the prediction of where and when such a bushfire starts. There is a wide range of typical causes, some of them are natural whereas some of them are related to human behavior. Natural causes involve *"lightning, volcanic eruption, sparks from rockfalls, and spontaneous combustion"* [2] and human involved ignitions originate e.g. from campfires, agriculture burns, accidents, machinery or arson [3].

That bushfires are a serious threat for regions with suitable climate conditions becomes apparent, when looking at the statistics of the last catastrophic bushfires in Australia. The so called *Black Saturday bushfires* burned an area of ca. 450 000 ha (which is ca. 1.75 times the area of the Saarland with ca. 257 000 ha) in just a few days during February 2009 in Victoria. The damage was immense with 173 fatalities, 414 injuries and more than 3500 structures destroyed. Thousands of people became homeless and more than 400 fires were recorded on Saturday 7th February 2009 in the Victorian region. [4]

Figure 1.1: A bushfire threatening a house[1]

[1] `http://www.bushfireinformation.com.au/wp-content/uploads/2012/11/Bushfire-Information.jpg` [last accessed Sept. 5, 2013]

1.2 Bushfire-Warn-Applications and Websites

To prevent fatalities and to enable appropriate decision-making by people located in bushfire-prone regions, informing the public early and understandably is a crucial aspect. Kunz et al. mention in their paper *"Understanding the Risk: Establishing of an Interactive System for the Visualization and Exploration of Natural Hazards and associated Uncertainties"* some important preconditions for a successful hazard response and crucial aspects are *"emergency management and planning, early warning, alerting of the population [and] strengthening of local knowledge"* [5]. These aspects are exactly the goal of *bushfire-warn-apps* for mobile devices and *bushfire-warn-websites* that emerged in the last years. They are available for many countries all around the world and especially popular in Australia.
They aim to provide information about the current bushfire situation to people located in bushfire-prone regions. Generally, such apps and websites provide a list of current bushfires that is based on publicly available bushfire data provided by the responsible authorities. In the recent past, these apps and websites started to provide information about the geographical location of the bushfires by displaying them on a map. Most of these apps additionally provide some special features like "warning sms" that are sent out in special cases and possibilities to upload pictures of the fires for other users.

Figure 1.2: Screenshot of the CFA Fire Ready app[2]

[2] http://cdn2.ubergizmo.com/wp-content/uploads/2013/02/FireReady-App-Screenshot1.jpg [last accessed Sept. 5, 2013]

When taking a closer look to how the fires are visualized in the publicly available apps and websites, it is remarkable, that almost all of them use the same visualization technique: a single iconic fire marker placed on the map. A few recent websites started to integrate a visualization of the current fire area; but up to now, none of the available apps provided a visualization other than a single marker per fire. The advantages of such simple visualizations are, that they are intuitive to interpret and easy to implement. But at the same time, they have a great disadvantage: the accuracy and the amount of valuable information that is communicated.
Of course, there are also bushfire visualizations that provide much more valuable information. These visualizations are typically used by professionals and expert users who are occupationally or scientifically concerned with these types of visualizations. But despite their advantage of containing much information, their drawback is that this information is encoded in the visualization and thus for laypersons not intuitively interpretable. Another important aspect here is, that they are generally not available to the public.
This bachelor thesis will compare both kinds of visualizations in a little more detail in the **Related Work** (3) section, but the most important outcome of the investigation is, that there is a gap between simple but less informative visualizations and informative but very complex visualizations.
It is this gap, that this thesis tries to fill by developing new approaches for the visualization of bushfires suitable for this kind of publicly available apps and websites and by evaluating them using a prototype implementation. The resulting visualizations aim to be easy to understand for laypersons while at the same time try to communicate more important information than the currently used visualizations.

Chapter 2
Goals

The following sections will define the goals of this thesis which can be divided into three main parts:

2.1 Concepts

The first goal is the development of several **concepts** for the visualization of bushfires and the development of **algorithms** implementing these concepts.

The target platform are publicly available mobile applications and websites that aim to inform the public about the current bushfire situation in their region. To be suitable for this field of application, the concepts should exploit the data available in mobile devices and websites. This means that the algorithms implementing the developed concepts should build upon the publicly available bushfire-data, GPS-data that is available in mobiles or street-data that is available online.
The developed visualization approaches should offer an alternative to the currently widespread simple marker-placement visualization and the perimeter visualization by fulfilling some important properties. They aim to :

- integrate information about the bushfire's spatio-temporal development
- integrate information about the user's position and environment
- be intuitive to interpret and easy to understand for laypersons
- provide more relevant information than current visualizations used in apps and websites

In this way, they are designed to fill the gap introduced in the last part of the **Related Work** section (3) between complex expert visualizations and the current less informative public approaches.

2.2 Implementation

The second part is the **implementation** of all developed visualization concepts and algorithms in *Java*[3].

The task is to develop a prototype system in Java, offering an environment for testing and demonstrating the developed algorithms and for comparing the resulting visualizations with the current visualization approaches. Moreover, the program should be able to generate visualizations suitable for the evaluation phase.

2.3 Evaluation

The last goal of the bachelor thesis is to **evaluate** the developed visualization approaches by conducting a study investigating the communicated threat of the different visualizations and their popularity.

In the study, users are confronted with different scenarios and for each visualization, the communicated threat should be assessed and eventually compared to the other visualizations'.
Additionally, the users' acceptance of the developed visualizations should be acquired and compared to the popularity of the currently used visualizations.
These results will then show, whether the developed approaches are a suitable alternative for public bushfire warn apps and websites.

[3] `http://www.java.com/en/` [last accessed Sept. 5, 2013]

Chapter 3
Related Work

This chapter will summarize findings of the related literature and it will shortly discuss the paper *Comparison of Techniques for Visualising Fire Behaviour* by Black et al. [6] which addresses the development of expert visualization techniques for bushfires. The last section will eventually look at the currently available visualizations for bushfires.

3.1 Important Aspects in the Related Literature

There are four types of literature related to this work: literature addressing visualization in general, literature about dynamic spatio-temporal data, literature about the visualization thereof as well as literature about the visualization of natural hazards.
Currently, there is not much literature that is especially concerned with the visualization of bushfires or wildfire behavior, but Black et al.'s paper falls into this category. This is the reason why it is discussed in a little more detail in the next section. The literature falling in the four categories mentioned above provides some important aspects and some useful insights in how visualizations work:

The Visualization - Pipeline

A very important concept is the so called *visualization pipeline* [7]. This pipeline concept describes how visualizations are generated and which stages they pass. Generally, all visualizations go through three main stages called

1. *Acquisition*

2. *Transformation*

3. *Visualization*

At first, the data underlying the visualization is assessed through measurements, observations or in the case of bushfires through infrared scans conducted by special planes overflighing bushfires in order to assess the perimeter and other data [4]. All this is done in the first stage: the acquisition phase.

After the acquisition, the acquired data is then input to the transformation stage. Here, the relevant information that should be visualized is recovered. This typically means that the raw data coming from the acquisition stage is filtered and transformed into appropriate data types and the relevant information is gained through computations.
In current bushfire visualizations in apps and websites, this stage is barely existent. Typically, a fire marker is placed at the point of ignition or the raw data of the perimeter is directly visualized without any transformation step that filters information or gains the most relevant information for the user. Instead current public visualizations use the "*direct depiction*" [8] of the fire data that is publicly available, as the investigation at the end of this chapter shows.
As a common transformation stage, the concept of a pre-processing phase conducting feature extraction is mentioned by Pang and the approaches introduced here will use such pre-visualization processing to gain data that is relevant for the user to answer particular questions. As Pang states, "*good visualizations are designed to answer particular questions*" [7] which is especially true for the visualization of natural hazards, as here, people using warn apps and websites are typically interested in certain features of the fires like the position of the burning frontier or the fire's movement in the last hours. Thus, visualizations should understandably illustrate them.

The result of the transformation stage is finally input to the visualization stage that generates a graphical depiction of the transformed data. This visualization is then shown to the user in order to communicate the desired information.

Categorization of dynamic spatio-temporal Data

A second important aspect for the visualization development is the categorization of the data that is visualized.
When considering data describing the development of large bushfires, one is concerned with two dimensions: the temporal dimension and the spatial dimension. The temporal dimension refers to the fire's development and change over time

[4] `http://www.airaffairs.com.au/scanning_operations.html` [last accessed Sept. 5, 2013]

while the spatial dimension refers to the spatial extent, location and transformation of the fire. These types of data are therefore called *dynamic spatio-temporal data*.
Spatio-temporal phenomena can be categorized as proposed by Connie Blok in the paper "*Monitoring Change: Characteristics of Dynamic Geo-spatial Phenomena for Visual Exploration*" [9] according to the types of changes that occur over time. Here, changes are categorized as:

- Existential Changes (e.g. appearance and disappearance)
- Mutation (e.g. on the attribute level)
- Movement (e.g. movement along a trajectory or boundary shifts)

In this way, several important properties of a spatio-temporal phenomenon are covered by this categorization. A bushfire for example would suddenly appear at the location of ignition, in the following it would mutate with varying speed (on the attribute level by changing heat or intensity of the fire) and it would perform boundary shifts and movements in the environment. At the end, it would slowly disappear again at locations different from the location of appearance.

Categorization of the User Type

After characterizing the type of the underlying data and phenomena, the type of user is to be characterized. In the related literature concerned with the visualization of natural hazards, four types of users are identified [7]:

- expert users
- policy makers, decision makers
- operational users
- casual users

Expert users are familiar with the data sets and they have a very good understanding of them. In the bushfire scenario, expert users are e.g. fire fighters and scientists concerned with similar data sets. In contrast, policy makers and decision makers, like e.g. the government, are people that are not directly familiar with the data sets but need a good understanding of it in order to act appropriately. Operational users are people that are only familiar with a special and limited set of visualizations and the group of casual users covers all people that don't have a strong technical knowledge of the data sets. They are not concerned with visualizations thereof on an every-day basis and they use visualizations of natural hazards for educational and informational purposes.
These four groups of people have different knowledge about the natural hazard's

data and they all have different backgrounds. Thus, they all have different questions that can hardly be answered all by the same visualization. As a consequence, one of Mr. Pang's results is that "*different stakeholders have different needs and uses for such information. A "one-size-fits-all" approach in hazard visualization may therefore not be the right approach.*" [7]. Expert users will need different visualizations than the public and this thesis focuses on visualizations suitable for public applications.
But what is the public most interested in, regarding hazard visualization ? According to Pang, it is typically the own safety, the safety of loved ones, the threat of own belongings and possible evacuation routes that can lead the users to safer places. Consequently, visualizations suitable for the public should try to integrate these aspects and they should aim to provide answers to related questions in order to inform the public appropriately.

The Visualization Tradeoff

The fourth important aspect mentioned in the literature addresses the question, in which way visualizations for expert users differ from visualizations for the public.
As already mentioned, they will most likely answer different questions and visualizations for expert users will typically provide a much higher level of detail and more specialized data sets will be displayed. At this point, a very important tradeoff steps in. It is the tradeoff *completeness of data vs. comprehensibility of data* [7] [10] which says that the more data is visualized at a time, the less comprehensible it generally becomes.
Since expert users are used to the concerned data sets, more information can be shown simultaneously, maybe even the complete data set is depicted and their expert knowledge will be required to decode the information that is encoded in the visualization. The expert visualizations will be more on the "completeness side" of this tradeoff than the public visualizations which will probably put more emphasis on the comprehensibility of the visualization.

Visualization of Uncertainty

Moreover, there exists a fifth frequently mentioned issue: the visualization of data with uncertainty.
In the general case, uncertainty affects the whole process in each stage: from unreliable sensors in the acquisition phase, through uncertainty that comes with predictions in the transformation phase to uncertainties that result from the discreteness of the visual output. Some of these uncertainties, typically those that can lead to significantly different final results, should be visualized in a way that communicates this uncertainty. The papers *"Understanding the Risk: Establishing of an Interactive System for the Visualization and Exploration of Natural Hazards and associated Uncertainties"* [5] and *"Visualizing Natural Hazard Data and*

Uncertainties" [11] by Melanie Kunz and the paper *"Visualizing Uncertainty in Natural Hazards"* [7] by Alex Pang address these issues and propose to use visual variables like transparency, fuzziness, color, saturation, blurriness, geometric objects, etc. to emphasize uncertainty in natural hazard visualizations.

3.2 Expert - Visualizations by Black et al.

The paper *Comparison of Techniques for Visualising Fire Behaviour* [6] by Black et al. is closely related to this thesis' topic since it discusses the development of three expert - visualization techniques and their evaluation. The visualizations developed in the paper focus on the user group of expert users and are designed to visualize the output of fire behavior models like the *McArthur Forest Fire Danger Meter* [5] and computer based simulations of bushfires.
The paper introduces the McArthur Forest Fire Danger Meter, a comparably easy to use system to predict fire behavior used by many fire fighters and authorities in Australia. It can make *"predictions for a fire's rate of spread, flame height, spotting distance [...] and a value for the Fire Danger Index (FDI)."* [6] based on input variables like wind, temperature, slope, humidity, rainfall and fuel.
Besides this, nowadays several computer based simulation systems for bushfires exist and most of these predictions need to visualize the results for expert users. The authors state that *"as the accuracy of input data used in the models is important,[...] so too is the way the data are presented to the user"* and introduce three visualization approaches for bushfire simulation results:

Name	Dimension	Topographic Map	Satellite Images
Topographic Map	2D	✓	✗
Elevation Model	3D	✓	✗
Elevation Model with Satellite Overlay	3D	✗	✓

Table 3.1: The Visualizations developed by Black et al.

The first approach, the *Digital Topographic Map* is a 2D visualization of the area using a topographic map and a street-overlay. Here, the fire's development is shown by rendering the fire's perimeter at specific moments in time and by showing an animation of the perimeter's movement over 5 hours.
The second approach is the *Digital Elevation Model* offering a 3D view on the area. Here, an abstract model of the local topography was generated and textured with a topographic map and the street-overlay used in the first visualization.
To generate an even more realistic visualization of the fire and its environment, the *Digital Elevation Model with Satellite Overlay* uses the same 3D model of the terrain as the second visualization, but this time textures it with satellite images

[5] `http://royalcommission.vic.gov.au/Documents/Document-files/Exhibits/WIT-004-001-0315.pdf` [last accessed Sept. 5, 2013]

and the street-overlay.
In all three visualizations, the fire and its perimeter are rendered as an orange area in the scene.

All three approaches are eventually evaluated by the authors in a user-study. For this, several expert users from the Victoria Department of Sustainability and Environment were invited to use a prototype implementation of the visualizations to perform several tasks. The prototype implementation displayed the GUI of an expert system and a window showing the scene with the tested visualization. Finally, the users should outline the strengths and weaknesses of each visualization approach.

Their results showed, that the 2D approach was preferred by many users for identifying the fire's spread and perimeter. It is a suitable approach for fire-managers familiar with the local area. The 3D view provided a good impression of the topography in the area which is especially important for fire-fighters not familiar with the area, but the perimeter and its movement were harder to spot. In addition, the last visualization informed about the vegetation and allowed to spot farmlands and other features of the area due to the satellite images used.

But the paper's results also show, that there is not a perfect visualization approach; it always depends on the users and their needs. Each visualization increased the knowledge about the situation. In an expert system, it would probably be best to offer a range of different visualizations and to allow switching between them.

Similar to Black et al.'s paper, this thesis will introduce and evaluate 7 visualization approaches in the 2D domain specifically designed for the group of casual users.

Figure 3.1: The three visualizations developed by Black et al. [6]

[6] pictures taken from [6]

3.3 Current Bushfire Visualizations

3.3.1 Public Visualizations

As briefly outlined in the **Motivation** section, public visualizations of bushfires can be found in bushfire warn applications for mobile devices and bushfire warn websites that use widespread map services like Google Maps or OpenStreetMap to visualize the fires on a map. Several of these apps and websites emerged in the last years, some of them are limited to specific locations and regions while others work globally.

Visualization in Apps

When investigating the currently available apps for bushfire prone regions around the world, it is noticeable that virtually all use the same way to show the fires on the map: they display one simple, static and iconic fire marker per fire at its location of ignition.
The advantage of such a simple visualization is, that it is easy to implement and intuitive to interpret for laypersons. But it can be questioned in terms of accuracy, especially when fires cover huge areas. In those cases, showing the location of ignition can make people underestimate the danger originating from the fire and even lull users into a false sense of security when fire frontlines are already far away from the original ignition point. In these situations, it would be good, if visualizations would adapt to the user's location and the fire's behavior. But up to now, this is not the case in mobile bushfire warn apps.

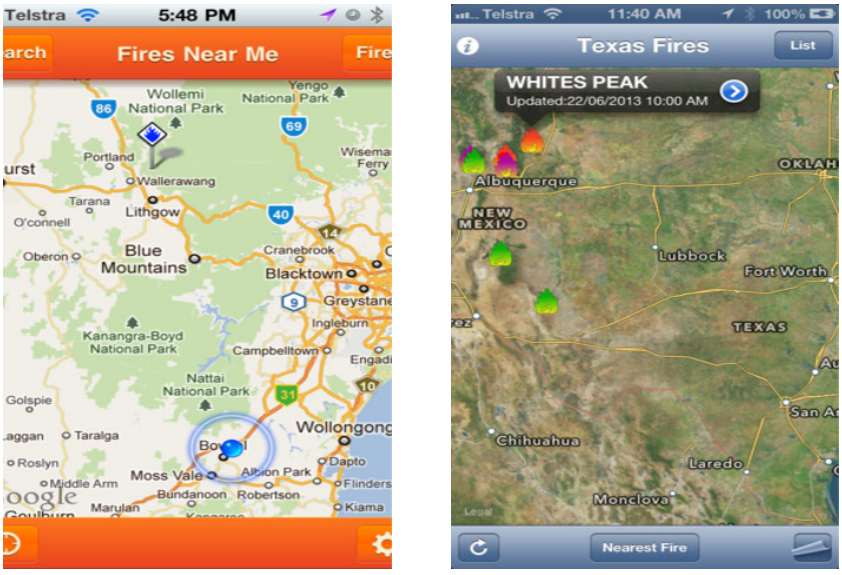

Figure 3.2: NSW Fires Near Me App (left) and TexasFires App (right) [7]

Considering the data available for mobile apps in smartphones or tablets, many improvements are possible. Typically, the online fire-data provided by the authorities consist of the current perimeter data and the point of ignition as well as some numerical data like burned area or status of the fire (going, controlled, etc.). Moreover, this data is updated in a regular fashion and thus, information about the spatio-temporal development of the fire is available, but currently not used in public visualizations.
Additionally, information that is not directly related to the fire can be used to augment the visualization and to personalize it. It would for example be adequate to integrate the user position in the visualization, either by visualizing the position of the user in different ways or by integrating information about the user's position into the transformation stage of the visualization that is used.
And it is not only the available GPS-data that suits for the usage in the transformation stage. Even street-data could be used, extrapolations of the fire behavior could be made or animations and other spatio-temporal visualization techniques could communicate much more valuable information about the fire's spread.

Used in Transformation Stage	Apps
Ignition Point	✓
Perimeter Data	✗
User Position	✗
Spatio-Temporal Development	✗
Extrapolation	✗
Street Data	✗
Animation	✗

Table 3.2: Current App - Visualizations

Visualization in Websites

Similar to bushfire warn apps, bushfire warn websites provide lists with current incidents and an online map showing the ongoing fires and planned burns. They typically provide less special features, most of them just display the bushfires on the map.
When investigating the visualizations used online, one will mainly see two approaches:
The vast majority of websites implements exactly the same visualization technique as corresponding apps do and place a single, static and iconic fire marker on the map at the location of ignition.
The rest implements a somewhat more informative visualization that renders the

[7] left: `https://itunes.apple.com/de/app/fires-near-me-nsw/id370891827?mt=8` [last accessed Sept. 5, 2013]
right: `https://itunes.apple.com/us/app/texas-fires/id426341846?mt=8` [last accessed Sept. 5, 2013]

available fire perimeter on the map. This means, the data provided by the authorities is displayed as a polygon showing the burned area. When the perimeter is updated, the old perimeter configuration is removed by the updated one.
But again, no real transformation takes place, the approach that is used in websites is more a *"direct depiction"* [8] of the raw data provided by the authorities and no spatio-temporal information about the fire's development or its dynamic is communicated, no street data is used, no animations are implemented and no personalized visualization (using the user's location) is currently available.

Used in Transformation Stage	Apps	Websites
Ignition Point	✓	✓
Perimeter Data	✗	(✓)
User Position	✗	✗
Spatio-Temporal Development	✗	✗
Extrapolation	✗	✗
Street Data	✗	✗
Animation	✗	✗

Table 3.3: Current Website - Visualizations

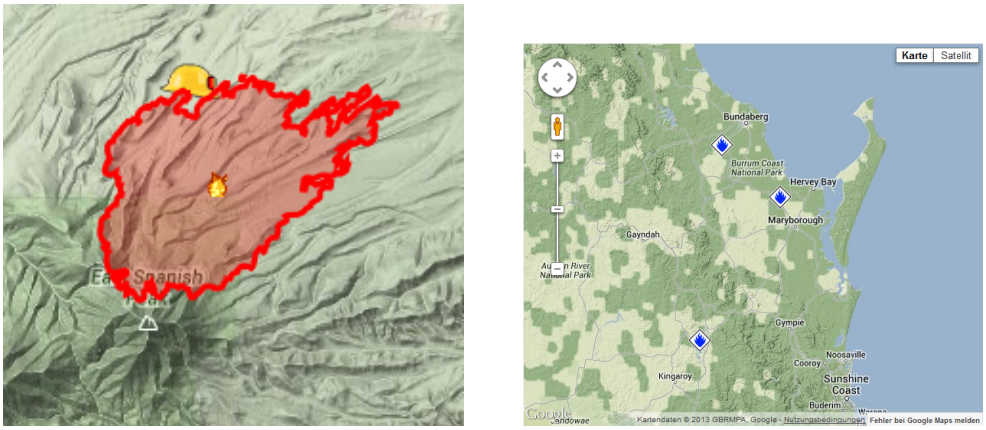

Figure 3.3: Google Crisis Map Screenshot (left) and Queensland Rural Fire Service Website Screenshot (right)[9]

[9]left: `http://google.org/crisismap/2013-nsw-bushfires` [last accessed Sept. 5, 2013]
right: `http://www.ruralfire.qld.gov.au/map.html` [last accessed Sept. 5, 2013]

3.3.2 Expert Visualizations

On the other side of the tradeoff between completeness and comprehensibility mentioned in the **Related Work** (3.1) section lies the expert visualization. These visualizations are targeting the group of experts and professionals like fire-managers and scientists that work with bushfire data on an every-day basis. As described in the previous section, expert visualizations generally communicate the results of complex bushfire simulations or other measured data sets. They are designed to deliver highly specialized information about flame height, smoke, fuel, fire intensity, heat spread and many other attributes in a way, that professionals can work with.
Expert visualizations consequently contain very much information about the fire; but this information is encoded in the visualization. The knowledge of an expert is required to properly understand the visualized data sets and generally, this interpretation is hardly possible or even impossible for laypersons.
Despite the fact that these visualizations and data sets are most often not available to the public, the visualizations itself are generally too complex and not intuitive understandable for casual users and thus not suitable for a public warn-application.

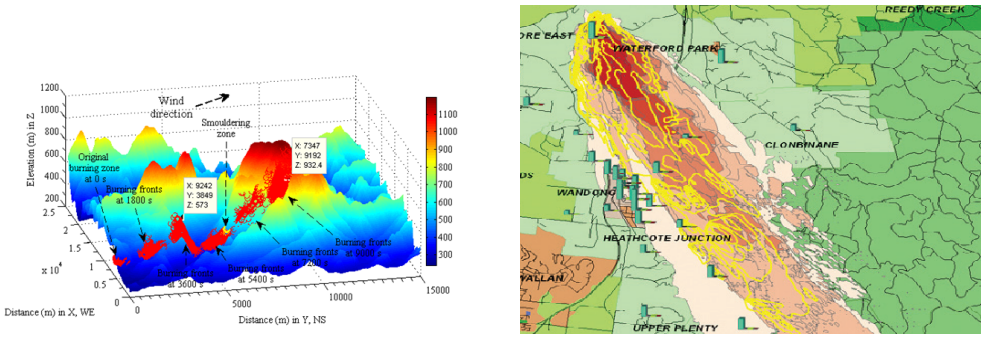

Figure 3.4: Scientific Visualization from [12] and Visualization from the FireDST Tool[11]

3.3.3 Conclusion

Summarizing one can say, that currently two types of visualizations exist: the complex expert visualizations that focus on the completeness part of the *completeness vs. comprehensibility* tradeoff and the public visualizations used in apps and websites that put emphasis on the comprehensibility while loosing accuracy and informative content.

The comparison shows that there is a "gap" between both approaches. To fill this "gap", new alternative visualizations for the public are developed in the

[11] picture taken from `http://www.bushfirecrc.com/sites/default/files/managed/resource/fire_note_109_high_res.pdf` [last accessed Sept. 5, 2013]

	Public Apps and Websites	Expert Systems
Pro	simple intuitive for laypersons	precise much information
Con	imprecise little information	complex unintuitive for laypersons not available to the public

Table 3.4: Current Visualizations - Summary

following chapter. These trade off comprehensibility and completeness in a way, that allows laypersons to gain more relevant information than offered by current public visualizations while being less complex than expert approaches. In addition to that, the new visualizations will include more information into the transformation stage than current public apps and websites do, in order to offer information that really matters for people endangered by bushfires.

Chapter 4
Concepts - The Visualization Algorithms

This chapter outlines the developed visualizations and algorithms. At first, a brief overview is given followed by a more detailed description of each algorithm and the pseudocodes.

4.1 Overview

"There are a number of different options available to view spatial information and selecting the right one or combination is essential for effective communication."
- Black et al. [6]

For this thesis, 7 visualization approaches for communicating information about ongoing bushfires were developed as well as the core algorithms computing the relevant information from the data available in apps and websites.
Following the concept mentioned in Pang's paper [7], all visualizations are designed to answer different important questions that people located in bushfire regions are confronted with.
For this, they generally use a pre-visualization stage (or transformation stage), in which they extract relevant features from the available data sets like special points of interest, the fire frontline, information about the development, spread and so on.
All in all, they aim to be alternatives for visualizations in warn apps and websites by trading off completeness of data and comprehensibility as described in the **Conclusion** (3.3.3) section of the previous chapter.
To be able to communicate more information while staying intuitive and inter-

pretable for laypersons, the approaches range from simple marker placements to more complex visualizations using colored streets, arrows, curves and areas.
The following table shows all new approaches compared to the current ones:
Abbreviations for the Algorithms:
NP = Nearest Point; **ND** = Nearest Point Development; **FF** = Fire Frontline;
FFD = Fire Frontline Development; **SP** = Spread; **DZ** = Danger Zone

Using	Apps	Web	NP	FF	ND	FFD	SP	DZ	Street
Ignition Point	✓	✓	✗	✗	✗	✗	✓	✗	✗
Perimeter Data	✗	(✓)	✓	✓	✓	✓	✓	✓	✓
User Position	✗	✗	✓	✓	✓	✓	✗	✗	✓
Spatio-Temp. Dev.	✗	✗	✗	✗	✓	✓	✓	✓	✗
Extrapolation	✗	✗	✗	✗	✗	✗	✗	✓	✗
Street Data	✗	✗	✗	✗	✗	✗	✗	✗	✓
Animation	✗	✗	✗	✗	✓	✓	✗	✗	✗

Table 4.1: Comparison of all Visualizations

The Fire - Geometry - Data

As defined in the **Goals** (2), the algorithms work with the bushfire data sets that are publicly available.
These data sets consist of a list of fires which is regularly updated via RSS feeds[12].
In these data sets, each *fire* itself consists of the *point of ignition* and a *perimeter* which is made up of a number of *geo-referenced polygons* edging the burning and burned areas.
To get the fire's development over time, an app or a website could save all incoming updates to maintain a *list of perimeters* per fire, each representing the fire at a different moment in time. By doing this, a set of discrete sample points in the temporal dimension exists, with which a visualization of spatio-temporal features becomes possible.

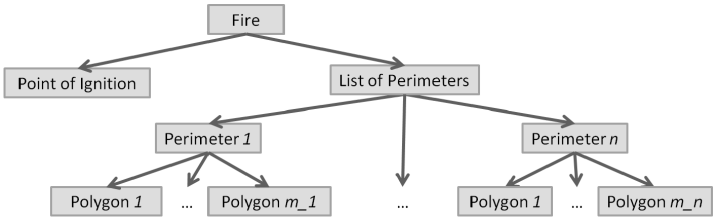

Figure 4.1: The Fire - Geometry - Data

[12] e.g. the Fire RSS feed by the CFA in Victoria: `http://www.cfa.vic.gov.au/rss-feeds/` [last accessed Sept. 5, 2013] or the NSW Rural Fire Service: `http://www.rfs.nsw.gov.au/dsp_content.cfm?cat_id=1358` [last accessed Sept. 5, 2013]

4.2 Nearest Point Algorithm

4.2.1 Motivation

The *Nearest Point Algorithm* is the first and the simplest new approach and falls into the category of typical marker-placement algorithms.
In contrast to the current placement at the point of ignition, which, as already outlined, is not a good representative of a bushfire, this algorithm aims to compute a marker-position that communicates valuable information to the user. For doing this, the algorithm takes into account the user's coordinates and the most up-to-date perimeter data of a fire to answer the question : *"How close is the fire to the user ?"*.

4.2.2 Algorithm

The perimeter data of a fire typically consist of multiple geo-referenced polygons as described in 4.1. If we now think of the algorithm's task in terms of 2D-geometry, the goal is to compute the point m on the polygons' outline, that has the minimal distance to the point u, representing the location of the user. This means, that the algorithm's main task is to solve the nearest point problem *"polygon - point"*
The algorithm computes this closest point m by considering the closest point to u on each of the polygons' line-segments connecting two adjacent vertices of a polygon. While iterating over all line segments of all polygons, the algorithm always saves the point that is closest to u in the variable $nearest$. This is a loop invariant that ensures that the algorithm delivers the correct result since at the end, all line-segments of the perimeter are visited and the point saved in the variable $nearest$ is returned.
To compute the nearest point on a line-segment, some vector calculations are used. When considering the line-segment connecting the points p_i and p_{i+1}, the algorithm firstly constructs the connection-vector $\vec{c} = p_{i+1} - p_i$ and the vector $\vec{s} = u - p_i$ going from the first vertex p_i to the user's position u. After that, the vector $\vec{s}$ is projected orthogonally onto $\vec{c}$ using the dot product $\vec{c} \cdot \vec{s}$. According to the dot product's definition,

$$\vec{c} \cdot \vec{s} = |\vec{c}||\vec{s}| \cos \alpha \tag{4.1}$$

where α denotes the angle between the two vectors $\vec{c}$ and $\vec{s}$, we can get the signed length l of the projection of $\vec{s}$ on $\vec{c}$ by

$$l = |\vec{s}| \cos \alpha = \frac{|\vec{c}||\vec{s}| \cos \alpha}{|\vec{c}|} = \frac{\vec{c} \cdot \vec{s}}{|\vec{c}|} \tag{4.2}$$

To easily get an information about where the projection of point u on the line going through p_i and p_{i+1} is located (relative to the part between p_i and p_{i+1}) the

algorithm considers the value

$$t = \frac{l}{|\vec{c}|} = \frac{\vec{c} \cdot \vec{s}}{|\vec{c}|^2} \tag{4.3}$$

Now we can distinguish three cases:

1. $t \leq 0$. In this case, the projection of u on the line is not between p_i and p_{i+1} and so, p_i is the closest point on the segment to u.
2. $0 < t < 1$. In this case, the projection of u on the line is between p_i and p_{i+1} and consequently the closest point to u on the segment is $p_i + t * \vec{c}$ since the closest point on a line to a point is the orthogonal projection of the point on the line.
3. $1 \leq t$. In this case, the projection of u on the line is not between p_i and p_{i+1} and p_{i+1} is the closest point on the segment to u.

This closest point is then compared to the closest point of all line-segments considered before ($nearest$) to check, whether it is closer to u than $nearest$. If this is the case, $nearest$ is overwritten. Finally, the algorithm returns a pair consisting of the closest point found and the distance from u to this point.
In addition, the algorithm uses the so called *orthodromic distance*[13] [13] when checking distances to respect the earth's round shape.

4.2.3 Pseudocode

See **Algorithm 1**.

4.2.4 Comments

This simple to compute and intuitive to understand placement enables the user to make decisions based on the marker placed on the map, because assuming the perimeters are updated in an appropriate frequency, the user can be sure, that there is no point closer than the marked one, that is threatening him.
In addition to showing the marker on the map, it might be a good idea to add information about the total distance. The study by Black et al. found, that *"both the numerical data and visualization graphics should be available"* because it can *"complement the visual output and [the user can] gain a better understanding of the fire situation."* [6]. Following this advice, one could add a text-note with the distance returned by the algorithm to the marker in order to make assessing the spatial relationship easier for the user.
In all following examples, the fire's perimeter is represented by the blue polygons and the user's position is indicated by the blue circle.

[13]The *orthodromic distance* is the shortest distance between two points on the surface of a sphere. Using a sphere with radius 6378 km approximates the distance of two points on the earth's surface

Algorithm 1 Nearest Point Algorithm

Input: u = the user's location
$perimeter$ = the fire's current perimeter
Output: Pair with nearest point to u on perimeter and corresponding distance d

$nearest = null; current = null; minDist = \infty; currentDist = \infty;$
for all polygons $polygon$ in $perimeter.polygons$ **do**
  $n =$ number of vertices in $polygon$
  for $i = 0 \rightarrow n - 1$ **do**
    $c = polygon.vertices[(i+1)\%n] - polygon.vertices[i]$
    $s = u - polygon.vertices[i]$
    $t = \frac{c \cdot s}{|c|^2}$
    if $t \leq 0$ **then**
      $current = polygon.vertices[i]$
    else if $0 < t < 1$ **then**
      $current = polygon.vertices[i] + t * c$
    else
      $current = polygon.vertices[(i+1)\%n]$
    end if
    $currentDist = orthodromicDist(u, current);$
    if $currentDist < minDist$ **then**
      $minDist = currentDist;$
      $nearest = current;$
    end if
  end for
end for
return $Pair(nearest, minDist)$

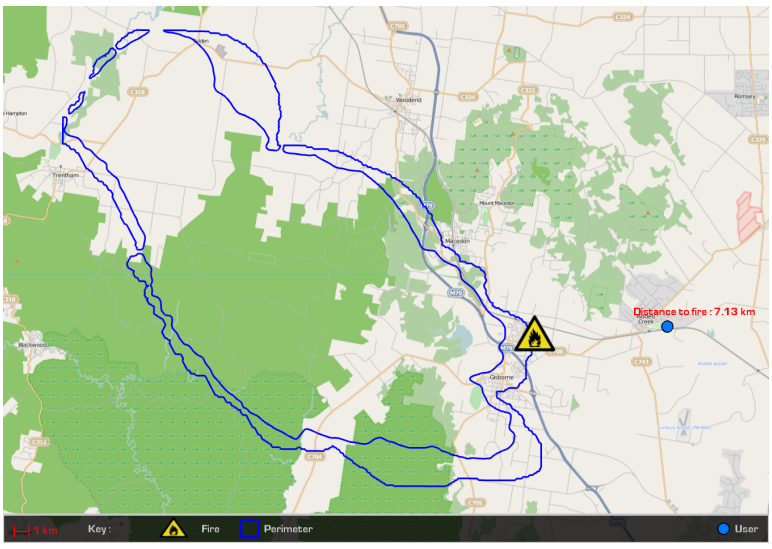

Figure 4.2: Example - Nearest Point Visualization

4.3 Fire Frontline Algorithm

4.3.1 Motivation

As mentioned in the literature concerning the visualization of natural hazards, the casual users are most concerned about their own and their family's safety [7]. This leads to the next question: *"What part of the bushfire threatens the user ?"*.
Since large bushfires are spreading in different directions with different velocities and burn intensively at the progressing frontline, the important part of such a fire is the actual fire-front that faces the user's position and moves towards him. Thus, the second algorithm is designed to approximate the fire-frontline facing and threatening the user. To visualize this approximation, it uses a method that goes further than just a marker placement and that is more spatial: an intuitive to interpret curve enabling the user to identify this relevant part of the fire.
The *Fire Frontline Algorithm*'s input is the most up-to-date perimeter of a fire and the user's coordinates as well as a parameter $maxPoints$ defining the length of the approximated frontline.

4.3.2 Algorithm

The algorithm approximates the relevant fire-front by computing an ordered list of geo-referenced control points that define the approximation. These control points are then passed to an implementation of a curve-visualization, for example a *Catmull-Rom-Spline* [14] as used in the prototype implementation, which smoothly connects all ordered control points from the beginning to the end. This smoothed curve through all control points is then an approximation of the fire-frontline and can be rendered on the map.

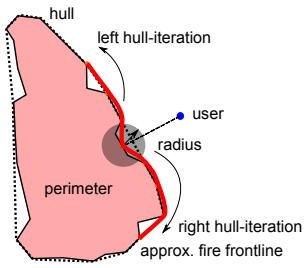

Figure 4.3: Sketch of the *Fire Frontline Algorithm*

The most interesting part of this visualization approach is the computation of the relevant control points. For this, the algorithm uses a hull of the fire's current perimeter, gained for example through the application of the *Graham-Scan-Algorithm* [15]. This algorithm computes the convex hull of the perimeter which is then saved in the perimeter object.
After computing the hull, the algorithm calls the *Nearest Point Algorithm* once to get the closest point on the perimeter to the user. This is the first control point and added to a list. After that, the algorithm starts iterating in over the hull's vertices to the left as well as to the right to find the next control points on the hull. Each two subsequent control points must have a minimum distance of $radius$. The while-loop eventually terminates when the iteration to the left and to the right "meet" on the other side of the hull or when $maxPoints$ many control points are found on the left and on the right side respectively. The while-loops at the very

end ensure, for consistency, that all returned lists have length $2 * maxPoints + 1$. The result of the algorithm after completion of all loops is the $controlPoints$ list defining the curve.

4.3.3 Pseudocode

Algorithm 2 depicts the function $computeNearestIndex$ used by the *Fire Frontline Algorithm* and **Algorithm 3** shows the pseudo-code of the algorithm itself.

Algorithm 2 computeNearestIndex

Input: u = the user's location
$perimeter$ = the fire's current perimeter
Output: index of the vertex in the perimeter's hull with minimum distance to u

```
1:  dist = ∞
2:  startIndex = 0
3:  for i = 0 → n − 1 do
4:      vertex = perimeter.convexHull.vertices[i]
5:      if ((currentDist = orthodromicDist(vertex, u)) < dist) then
6:          dist = currentDist
7:          startIndex = i
8:      end if
9:  end for
10: return startIndex
```

4.3.4 Comments

It is a widespread concept to color visualizations of natural hazards according to the phenomena represented by them. In their paper, Kunz et al. for example used the colors gray, blue and purple to visualize snow avalanche parameters because these "*cold colors [are] reflecting the characteristics of snow*" [11]. Consequently it is certainly a good idea to use hot colors like red, orange and yellow to color the fire-frontline and to intuitively give a "fire" and "danger" feeling to it. Furthermore, the nearest control point can be marked by a big fire-marker and each other control point by a small one. When implemented in an app or website, it is not necessary to display the current perimeter's polygons on the map when using this visualization approach. Just rendering the fire front should suffice to answer the question *"What part of the bushfire threatens the user ?"*.

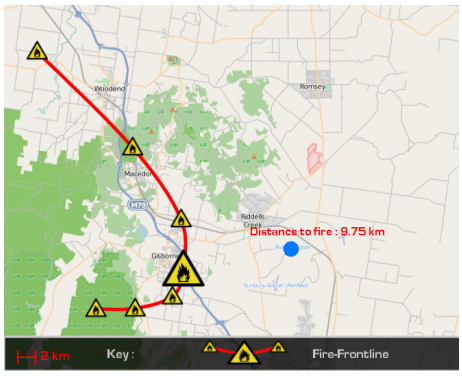

Figure 4.4: Example - Fire Frontline Visualization

Algorithm 3 Fire Frontline Algorithm

Input: u = the user's location
$perimeter$ = the fire's current perimeter
$maxPoints$ = the maximum number of control points per direction
Output: a list with $2 * maxPoints + 1$ controlpoints for a curve approximating the frontline

1: $controlPoints$ = empty list of locations; $lPoints$ = 0; $rPoints$ = 0;
2: $n = perimeter.convexHull.vertices.size()$
3: $realNearestPoint = NearestPointAlgorithm(u, perimeter)$.first
4: $controlPoints.add(realNearestPoint)$
5: $lastL = realNearestPoint$; $lastR = realNearestPoint$
6: $radius = computeAppropriateRadius(realNearestpoint, u)$
7: $nearestIndex = computeNearestIndex(u, perimeter)$
8: $indexL = (nearestIndex - 1 \geq 0)?nearestIndex - 1 : n - 1$
9: $indexR = (nearestIndex + 1)\%n$
10:
11: **while** $indexR \neq indexL$ **do**
12: $vertexL = perimeter.convexHull.vertices[indexL]$
13: $vertexR = perimeter.convexHull.vertices[indexR]$
14: **if** $rPoints < maxPoints$ **and** $orthodromicDist(vertexR, lastR) \geq radius$ **then**
15: $controlPoints.addLast(vertexR)$
16: $lastR = vertexR$
17: $rPoints++$
18: **end if**
19: **if** $lPoints < maxPoints$ **and** $orthodromicDist(vertexL, lastL) \geq radius$ **then**
20: $controlPoints.addFirst(vertexL)$
21: $lastL = vertexL$
22: $lPoints++$
23: **end if**
24: **if** $lPoints == maxPoints$ **and** $rPoints == maxPoints$ **then**
25: **break**
26: **end if**
27: $indexR = (indexR + 1)\%n$
28: **if** $indexR == indexL$ **then**
29: **break**
30: **end if**
31: $indexL = (indexL - 1 \geq 0)?indexL - 1 : n - 1$
32: **end while**
33:
34: **while** $lPoints < maxPoints$ **do**
35: $controlPoints.addFirst(lastL)$
36: $lPoints++$
37: **end while**
38: **while** $rPoints < maxPoints$ **do**
39: $controlPoints.addLast(lastR)$
40: $rPoints++$
41: **end while**
42: **return** $controlPoints$

4.4 Nearest Point Development Algorithm

4.4.1 Motivation

The *Nearest Point Algorithm* and the *Fire Frontline Algorithm* as described so far both use the most up-to-date perimeter data available in the application to statically visualize the current situation. But they don't provide any information about the temporal development and the spread dynamic of the fire. The following two algorithms are extending the nearest point and the fire frontline approach by adding the temporal dimension to the visualization. They do so by providing a static way to visualize the fire's spatio-temporal development towards the user, as well as an animated way, showing an approximation of the development in the last hours.
In this sense, the *Nearest Point Development Algorithm* answers the question : "*How did the fire develop towards the user's location ?*". The inputs are the fire's chronologically ordered list of all known perimeters in the past, the user's coordinates and some parameters defining the smoothness of the animation ($iFrames$), the amount of samples used in the algorithm ($timeSteps$) and the sampling frequency ($timeInterval$).

4.4.2 Algorithm

The *Nearest Point Development Algorithm* is based on a commonly used technique to deal with temporal data: sampling in the temporal dimension [8]. Concretely, the algorithm chooses the current perimeter and some perimeters at discrete moments in the past and applies the *Nearest Point Algorithm* to them. The resulting locations are then statically or dynamically visualized. For the prototype implementation, two versions of the algorithm were implemented:

(1) The first algorithm implements a static visualization of the development. The algorithm computes the chronologically ordered list of nearest points and places a marker at each location. Markers representing locations in the past are increasingly transparent and each two subsequent markers are connected by an arrow. The resulting visualization provides information about the current location of the nearest point and about the development of the fire, especially the spreading speed (when taking the time-span between two subsequent markers and the map's scale into account) in the user's direction.

(2) The second algorithm generates a frame-by-frame animation of the development. For this, the algorithm also applies the *Nearest Point Algorithm* multiple times in order to get the results at some discrete moments in time. Unfortunately, the perimeter-updates are typically not as frequent as necessary for giving the impression of a fluent movement when showing them frame by frame in an animation. Thus, the algorithm interpolates between two

subsequent moments in time to generate more frames filling the "gaps" between distant marker positions. The interpolation fineness is controlled by the parameter $iFrames$ defining the number of frames computed between two successive steps in the temporal development. The resulting frames are then ordered chronologically and can be rendered consecutively with the desired animation speed. The result shows the movement of the nearest point towards the user in the last hours and gives an impression of the fire's speed in the user's direction. To emphasize the current location of the fire, the last frame (showing the current nearest point) is to be rendered for a longer time span. This makes it easier for the user to understand the development and to recognize the current situation.

4.4.3 Pseudocode

Algorithm 4 depicts the static version, **Algorithm 5** shows the animated version and **Algorithm 6** is used by the latter for interpolation.

Algorithm 4 Nearest Point Development Algorithm (static)

Input: u = the user's location
$fire$ = the fire-data containing the chronological list of all perimeters
$timeInterval$ = time interval between two temporal sampling points in hours
$timeSteps$ = number of sampling steps
Output: chronologically ordered list of marker positions,
two successive markers should be connected by an arrow
markers in the past should be increasingly transparent

1: $list$ = empty list
2: **for** $i = timeSteps \rightarrow 0$ **do**
3: $perimeter = getPerimeterXHoursAgo(fire, i * timeInterval)$
4: $nearest = NearestPointAlgorithm(u, perimeter).first$
5: $list.add(nearest)$
6: **end for**
7: **return** $list$

4.4.4 Comments

Bringing the temporal dimension to the visualization of bushfires is a very important aspect. Two ways to do this are outlined here, both using simple marker placements, once in conjunction with connecting arrows emphasizing the development over time and once animated. To make the visualization more comprehensible, the corresponding timestamps should be rendered next to the markers and the key of the map should indicate, which time-span is represented by an arrow. In the animated version, an information about which time-span is animated (e.g. last 5 hours) should be given.

Algorithm 5 Nearest Point Development Algorithm (animated)

Input: u = the user's location
$fire$ = the fire-data containing the chronological list of all perimeters
$timeInterval$ = time interval between two temporal sampling points in hours
$timeSteps$ = number of sampling steps
$iFrames$ = the number of interpolation frames between two sampling steps
Output: an animation consisting of $(iFrames + 1) * timeSteps + 1$ many frames showing the nearest point's development over time

$frameList$ = empty list
for $i = timeSteps \rightarrow 0$ **do**
  $perimeter = getPerimeterXHoursAgo(fire, i * timeInterval)$
  $nearest = NearestPointAlgorithm(u, perimeter).first$
  if $i < timeSteps$ **then**
    $from = frameList.getLastMarkerPosition()$
    $to = nearest$
    for $t = 1 \rightarrow iFrames$ **do**
      $iPos = interpolateLocation(from, to, \frac{t}{iFrames+1})$
      $frameList.add(MarkerVisualizationOf(iPos))$
    end for
  end if
  $frameList.add(MarkerVisualizationOf(nearest))$
end for
$animation$ = new frame by frame animation with frame-sequence $frameList$

return $animation$

Algorithm 6 interpolateLocation

Input: $from$ = start location
to = end location
$timeFraction$ = fraction of the way between,
generally $0 \leq timeFraction \leq 1$
Output: the interpolated location between $from$ and to

$c = to - from$
$interpolation = from + timeFraction * c$
return $interpolation$

Figure 4.5: Example - Nearest Point Development Visualization

4.5 Fire Frontline Development Algorithm

4.5.1 Motivation

The underlying idea for the *Fire Frontline Development Algorithm* is the same as in the *Nearest Point Development Algorithm*: adding the temporal dimension to the visualization. This time, the spatio-temporal development of the fire frontline facing the user is visualized. Again, a static visualization with connecting arrows as well as an animation are suitable and both are described in the pseudocode beneath.
Here, the inputs are the same as in the *Nearest Point Development Algorithm*. The question that is answered by the visualization is "*How did the relevant fire frontline move towards the user ?*".

4.5.2 Algorithm

The fire frontline equivalent to the *Nearest Point Development Algorithm* works very similar to its marker-placement counterpart. The algorithm's parameters are the same as in the *Nearest Point Development Algorithm* except that one further parameter $maxPoints$, known from the *Fire Frontline Algorithm*, is added. Again, two visualizations exist:

1. The visualization in the static version renders chronologically ordered fire frontlines, where old frontlines are rendered transparent and two subsequent frontlines are connected by an arrow going from one nearest point to the nearest point in the next time step.
   The most important difference to the *Nearest Point Development Algorithm* is, that the algorithm is concerned with multiple positions at each time step. These positions represent the frontline's control points at a specific moment in time.
   To handle this, the static algorithm returns a list of lists. Each of those lists contains the frontline's control points at the respective time step. When for example accessing the returned list at [timeSteps][0], the leftmost control point of the most up-to-date fire-frontline is returned.

2. The animated version of this approach is a frame by frame animation showing the movement of the fire-frontline towards the user. Here, the same problem as in the *Nearest Point Development Algorithm* occurs: The perimeter update intervals are usually too big to give the impression of a fluent movement, especially when the fire frontline progressed fast towards the user. In this case, the interpolation approach works as well. The algorithm computes new control points by interpolating between two control points in two subsequent frontlines. This is done with a further for-loop in line 10 of the pseudocode. Here, the fact that all control point lists returned

by the *Fire Frontline Algorithm* called in line 4 have the same length of $2 * maxPoints + 1$ comes in handy. It ensures, that each control point has an "interpolation partner" in the next time step's frontline. Eventually, the last frame containing the current frontline should be displayed for a longer time-span to emphasize the current situation.

4.5.3 Pseudocode

Algorithm 7 Fire Frontline Development Algorithm (static)

Input: u = the user's location
$fire$ = the fire-data containing the chronological list of all perimeters
$maxPoints$ = the maximum number of control points per direction
$timeInterval$ = time interval between two temporal sampling points in hours
$timeSteps$ = number of sampling steps
Output: chronologically ordered list of control point lists defining the fire frontlines

1: $list$ = empty list
2: **for** $i = timeSteps \rightarrow 0$ **do**
3: $perimeter = getPerimeterXHoursAgo(fire, i * timeInterval)$
4: $controlPoints = FireFrontlineAlgorithm(u, perimeter, maxPoints)$
5: $list.add(controlPoints)$
6: **end for**
7: **return** $list$

4.5.4 Comments

Analogue to the *Nearest Point Development Algorithm*, the corresponding timestamps should be visualized, the time-span an arrow represents should be communicated and the time-span the animation displays should be noted for the viewer.
Especially for publicly available visualizations of natural hazards like bushfires, it is, as Kunz et al. states, very important that the *"underlying data [is] presented to decision makers in an understandable and interpretable way."* [5]. That is why an application implementing any of the introduced visualiza-

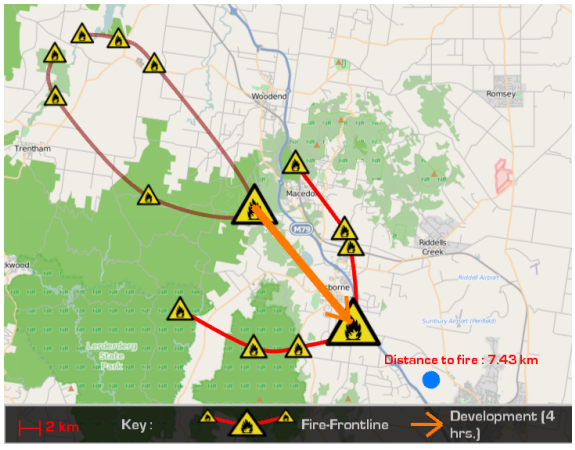

Figure 4.6: Example - Fire Frontline Development Visualization

Algorithm 8 Fire Frontline Development Algorithm (animated)

Input: u = the user's location
$fire$ = the fire-data containing the chronological list of all perimeters
$maxPoints$ = the maximum number of control points per direction
$timeInterval$ = time interval between two temporal sampling points in hours
$timeSteps$ = number of sampling steps
$iFrames$ = the number of interpolation frames between two sampling steps
Output: an animation consisting of $(iFrames + 1) * timeSteps + 1$ many frames showing the frontline's development over time

1: $frameList$ = empty list
2: **for** $i = timeSteps \rightarrow 0$ **do**
3: $perimeter = getPerimeterXHoursAgo(fire, i * timeInterval)$
4: $controlPoints = FireFrontlineAlgorithm(u, perimeter, maxPoints)$
5: **if** $i < timeSteps$ **then**
6: $fromPoints = frameList.getLastControlPoints()$
7: $toPoints = controlPoints$
8: **for** $t = 1 \rightarrow iFrames$ **do**
9: $iPoints$ = empty list
10: **for** $j = 0 \rightarrow maxPoints - 1$ **do**
11: $iCP = interpolateLocation(fromPoints[j], toPoints[j], \frac{t}{iFrames+1})$
12: $iPoints.add(iCP)$
13: **end for**
14: $frameList.add(Curve - Visualization(iPoints))$
15: **end for**
16: **end if**
17: $frameList.add(Curve - Visualization(controlPoints))$
18: **end for**
19: $animation$ = new frame by frame animation with frame-sequence $frameList$
20: **return** $animation$

tions should attach importance to additionally displaying an understandable and appropriate key.
As a result, perceiving the movement with this visualization through arrows and corresponding timestamps or even animations communicates much more information relevant for the process of decision making than visualizations that do not include the temporal development.

4.6 Spread Algorithm

4.6.1 Motivation

The *Spread Algorithm* is designed to improve the perimeter visualization used in some websites by providing not only a spatial visualization of the current fire status but by providing a spatio-temporal visualization. The user should receive an impression of how the bushfire came to its current form. This is done by adding the temporal dimension to the perimeter depiction in form of arrows with variable length and thickness. Concretely, a bundle of arrows starting at the fire's point of ignition provide information about the fire's dynamic by indicating the total extent of the fire's progress (measured from the point of ignition on) and its average spread speed during the last hours in different directions . The motivation for the visualization design itself comes from the classical "compass-rose" [14]. Moreover, the visualization is independent from the user's position and can thus be used in devices or applications that do not have access to the user's coordinates.
The question concerned by this algorithm is consequently: *"How did the fire spread?"* and the input for the algorithm is the fire's chronologically ordered list of perimeters and its point of ignition as well as the number of arrows used (num) and the time-span in the past that is taken into account (h hours) for computing the average spread speed.

4.6.2 Algorithm

The *Spread Algorithm* uses a 2D ray-casting approach to compute the relevant information. The parameter num defining the number of arrows also defines the amount of rays that are generated and shot. The directions are computed by dividing all four cardinal directions by the amount of rays shot, so that each ray has an rad-angle of $\frac{2\pi}{num}$ to the adjacent ones.
After a ray is generated, it is then shot from the point of ignition to the computed direction and the last intersection point with the perimeter that was up-to-date h hours ago is determined ($interPast$) as well as the last intersection point with the current perimeter ($interNow$). For the ray-casting,

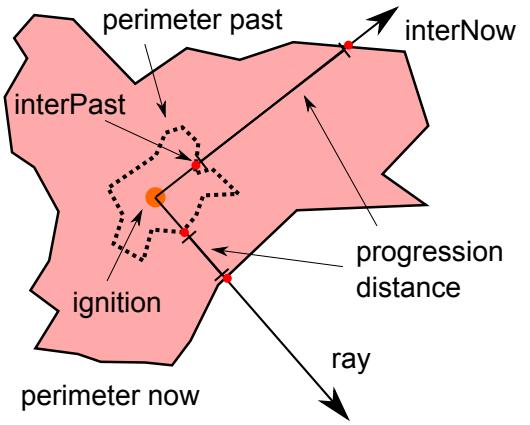

Figure 4.7: Sketch of the 2D Ray-Casting

[14] http://en.wikipedia.org/wiki/Compass_rose [last accessed Sept. 5, 2013]

a method $shootPerimeter$ is called, delegating the intersection-computation to all its polygons by using a method $shootPolygon$. This method then tests for intersection with each of the polygon's line-segments by using standard 2D-ray-casting algorithms like the fast 2D version of the line-segment intersection computation by Goldman[15] [16]. If a ray does not hit the perimeter, e.g. because the perimeter is not closed, the intersection with the convex hull is computed instead.
As a result, the fire's extent (in the ray's direction) in the past and in the present is given by the intersection points. This already defines the length of the arrow, which goes from the point of ignition to the last intersection with the current perimeter. The thickness of the arrow is computed in the following by using a function of the average spread speed in the ray's direction in the last h hours. This speed can be approximated by the distance the fire progressed in the ray's direction divided by the time it took the fire to progress. Consequently, the average spread speed in the direction of ray i during the last h hours is

$$v_i = \frac{orthodromicDist(interPast_i, interNow_i)}{h} \tag{4.4}$$

This speed is then mapped to an appropriate thickness for the arrow where fast progress is mapped to thick arrows and slow progress to thin ones.
The final result of the algorithm is then the list of all arrows.

4.6.3 Pseudocode

Algorithm 10 depicts the main *Spread Algorithm* using **Algorithm 9** and **Algorithm 11** for the ray-casting. **Algorithm 12** is a standard 2D ray-casting algorithm by Goldman [16] used for the intersection computation between the ray and a line segment.

Algorithm 9 shootPerimeter

Input: $origin$ = the point where the ray starts
dir = the ray's direction
$perimeter$ = the perimeter
Output: the point where the ray intersects the perimeter (with max. distance to $origin$)

```
1: intersections = empty list
2: for all Polygon poly in perimeter.allPolygons do
3:     intersections.add(shootPolygon(origin, dir, poly))
4: end for
5: return getMostDistantPoint(origin, intersections)[16]
```

[15] the pseudocode thereof is depict in **Algorithm 12**. Here, the 2D vector cross-product is defined as: $a \times b = a_x * b_y - a_y * b_x$

[16] The function $getMostDistantPoint$ straightforwardly determines the point within a set of intersection points, which has the greatest distance to the ray's origin.

Algorithm 10 Spread Algorithm

Input: $fire$ = the fire-data containing the chronological list of all perimeters
p = the fire's point of ignition
num = the number of arrows
h = the hours in the past concerned for the spread speed computation

Output: a list of num arrows, each with an rad-angle of $\frac{2\pi}{num}$ to the adjacent ones, the arrows' length indicates the total spread extent in the particular direction, the arrows' thickness indicates the spread speed in the last h hours in the particular direction

1: $arrows$ = empty list
2: $alphaStep = \frac{2\pi}{num}$
3: $perimeterPast = getPerimeterXHoursAgo(fire, h)$
4: $perimeterNow = getPerimeterXHoursAgo(fire, 0)$
5: **for** $i = 0 \rightarrow num - 1$ **do**
6: dir = new Vector($cos(i * alphaStep), sin(i * alphaStep)$)
7: $interPast = shootPerimeter(p, dir, perimeterPast)$
8: **if** $interPast = NULL$ **then**
9: $interPast = shootPolygon(p, dir, perimeterPast.convexHull)$
10: **end if**
11: $interNow = shootPerimeter(p, dir, perimeterNow)$
12: **if** $interNow = NULL$ **then**
13: $interNow = shootPolygon(p, dir, perimeterNow.convexHull)$
14: **end if**
15: $thick = computeThickness(\frac{orthodromicDist(interPast, interNow)}{h})$
16: $arrow$ = new Arrow($p, interNow, thick$)
17: $arrows.add(arrow)$
18: **end for**
19: **return** $arrows$

4.6.4 Comments

This visualization approach augments the static perimeter visualization by intuitively presenting information about a fire's spread dynamic. Long and thick arrows represent forceful fire progression, whereas short and thin ones represent a less rushing fire movement.
Implemented in a warn - application, the amount of arrows may vary. For big fires for example, it is probably good to generate many arrows covering many directions to properly inform people around the fire.

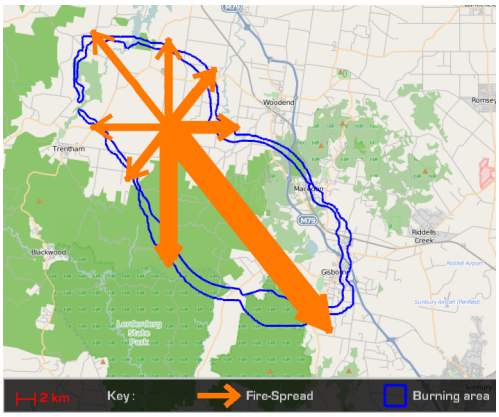

Figure 4.8: Example - Spread Visualization

Algorithm 11 shootPolygon

Input: $origin$ = the point where the ray starts
dir = the ray's direction
$poly$ = the polygon
Output: the point where the ray intersects the polygon (with max. distance to $origin$)

1: $intersections$ = empty list, $vertices = poly.allVertices, n = vertices.size$
2: **for** $i = 0 \rightarrow n - 1$ **do**
3: $c = vertices[(i + 1)\%n] - vertices[i]$
4: $t = computeIntersectionTime(vertices[i], c, origin, dir)$
5: **if** $0 \leq t \leq 1$ **then**
6: $intersections.add(vertices[i] + t * c)$
7: **end if**
8: **end for**
9: **return** $getMostDistantPoint(origin, intersections)$

Algorithm 12 computeIntersectionTime

Input: $originLine$ = the point where the line to intersect starts
$dirLine$ = the vector along the line
$origin$ = the point where the ray starts
dir = the ray's direction
Output: a time t, so that $originLine + t * dirLine$ intersects the ray $origin + u * dir$ for some $u > 0$

1: $u = \frac{originLine \times dirLine - origin \times dirLine}{dir \times dirLine}$
2: **if** $u \leq 0$ **then**
3: **return** ∞
4: **end if**
5: $dirLineXdir = dirLine \times dir$
6: **if** $dirLineXdir = 0$ **then**
7: **return** ∞
8: **end if**
9: **return** $\frac{origin \times dir - originLine \times dir}{dirLineXdir}$

4.7 Danger Zone Algorithm

4.7.1 Motivation

The next algorithm is based on the fact, that the danger originating from the fire is not distributed equally in all directions as might be the case in other natural hazards like e.g. radiation catastrophes. To declare the dangerous areas around the fire, it would be inappropriate to just draw circles around the point of ignition. Instead, bushfire danger zones are aligned around the fire perimeter. Thus the *Danger Zone Algorithm* determines zones in the fire's environment that indicate the potential future location of the bushfire. These zones are independent from the user's location and eventually rendered on the map.
To determine these zones in the transformation stage, three approaches were developed. The first one is to declare danger zones that have constant distances to the fire's current perimeter hull, independent from the direction. The second one extrapolates the fire's movement using information about the fire's spread in the past. This extrapolation can of course not replace a bushfire simulation, but it can help to inform about the fire's movement by using the publicly available information about the geometric fire progression. The third approach is to use a hybrid algorithm that merges both approaches and extrapolates the fire's movement, but guarantees a minimum danger zone distance to the fire.
Input for all algorithms is again the chronologically ordered list of perimeter data and some parameters that define the number of zones to compute ($maxZones$), the minimum buffer distance ($bufDist$), the type of algorithm used ($mode$) and extrapolation parameters (h and $extraHrs$). In summary, the question addressed by this visualization is: *"Where could the fire be in the future ?"*.

4.7.2 Algorithm

All following algorithms are based on the same approach: conducting an *outwards-polygon-buffering* on the current perimeter's hull. As described in Section 4.3.2 (*Fire Frontline Algorithm*), the hull can be computed with algorithms like the *Graham-Scan-Algorithm* [15] and is saved in the perimeter object. The main difference between the three developed approaches is how the polygon is buffered:

1. In the simple version ($mode$ 0), a *constant outwards-polygon-offsetting* is implemented that offsets each line segment of the hull orthogonally outwards to gain a constant buffer zone around the current fire area. By doing this, the intersection point of two adjacent line segments after offsetting is determined using the function $intersectLL$ (intersect-Line-Line) and represents one vertex of the buffer zone polygon (see Figure 4.10). To gain the whole buffer zone polygon, all line-segments are offset and all intersections of neighboring lines are added as vertices.
   This is done with different buffer distances in order to generate multiple

danger zone layers. All resulting polygons are eventually saved in an 2D-array with $maxZones$ rows, each row containing the vertices of a danger zone polygon.

2. The second approach ($mode$ 1) uses extrapolation, a useful concept for the visualization of dynamic geo-spatial phenomena, as mentioned in [9]. Here, the position of the fire hull's vertices is extrapolated under the assumption, that the fire will locally continue spreading with similar direction and speed as in the near past. Consequently, the approach implements *variable offsetting* and the resulting danger zone is buffered differently at each vertex. The computation works as follows: The algorithm looks at each vertex on the current perimeter's hull ($vertex_i$) and computes its future position by extrapolation. Two parameters are necessary for this: the local spread direction and the local average spread speed.
At first the (normalized) direction is approximated by computing a so called anchor point ($anchor_i$) using the function $computeAnchor$. The direction from this anchor point to the current vertex is an approximation for the local spread direction.

$$dir_i = \frac{vertex_i - anchor_i}{|vertex_i - anchor_i|} \tag{4.5}$$

The implementation of $computeAnchor$ can be different. In the prototype, the nearest points to $vertex_i$ on past perimeters are computed using the *Nearest Point Algorithm*. The anchor is then defined as the center point of the resulting 2D point cloud.
After approximating the local spread direction, the local spread speed is approximated by using an approach similar to the one used in the *Spread Algorithm*. Here, the nearest point of $vertex_i$ to the perimeter h hours ago ($nearest_i$) is determined and the average spread speed of the last h hours is computed as the division

$$v_i = \frac{orthodromicDist(vertex_i, nearest_i)}{h} \tag{4.6}$$

Having both an approximation for the direction and the speed, the extrapolated position of the vertex for the next t hours is defined as

$$ex_i = vertex_i + t * v_i * dir_i \tag{4.7}$$

Extrapolating each vertex of the current hull in this way yields the buffered danger zone polygon depicting the t - hour extrapolation of the fire (see sketch in Figure 4.11).

3. The hybrid approach ($mode$ 2) simply combines the advantages of the latter two approaches: it extrapolates the fire but at the same time guarantees a minimum buffer zone distance. This is achieved by extrapolating each vertex as in the second algorithm, but after extrapolation, the distance of

the new extrapolated position and the original hull is determined. If this distance is smaller than the threshold $bufDist$, the constant offsetting of the first algorithm is applied to the line segments surrounding $vertex_i$.

4.7.3 Pseudocode

Algorithm 13 shows the pre-processing and initialization stage of the main *Danger Zone Algorithm* shown in **Algorithm 14**

Algorithm 13 Danger Zone Algorithm - <pre-processing and initialization>

1: $hullV = fire.currentPerimeter.convexHull.allVertices, n = hullV.size$
2: $vectors =$ new Vector[n], $normals =$ new Vector[n]
3: $offsetV =$ new Vector[$maxZones$][$2 * n$], $result =$ new Point[$maxZones$][n]

4: **for** $i = 0 \rightarrow n - 1$ **do**
5: $vectors[i] = hullV[(i + 1)\%n] - hullV[i]$
6: **end for**
7: **for** $i = 0 \rightarrow n - 1$ **do**
8: $norm =$ new Vector($-vectors[i].y, vectors[i].x$).$normalize$
9: **if** $norm \cdot vectors[(i + 1)\%n] \geq 0$ **then**
10: $norm = -norm$
11: **end if**
12: $normals[i] = norm$
13: **end for**

4.7.4 Comments

This approach uses colored areas to illustrate the dangerous regions around the bushfire. The colors of choice are again hot color interpolations between red and yellow to indicate the fire threat. Additionally, when using extrapolations, uncertainty will always play a role. As a consequence, this uncertainty should be mentioned in the key and be visualized by areas that become increasingly transparent with increasing forecast timespan t, especially when multiple danger zones are displayed.

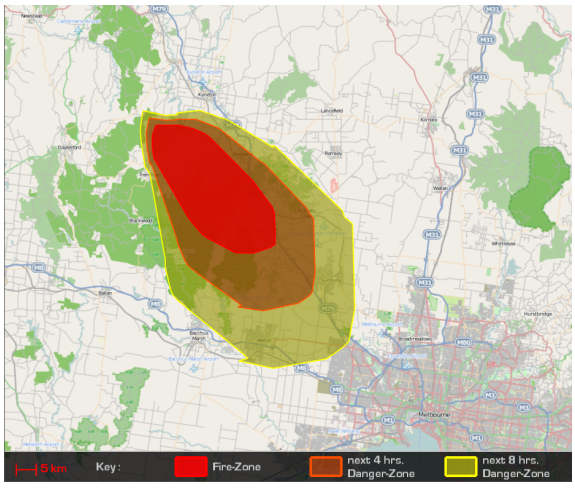

Figure 4.9: Example - Danger Zone Visualization

Algorithm 14 Danger Zone Algorithm

Input: $fire$ = the fire data containing the list of perimeters
$bufDist$ = the (minimum) buffer distance (only mode 0 and 2)
$maxZones$ = the number of zones to generate
h = using average spread speed of last h hours (only for mode > 0)
$extraHrs$ = extrapolation time-steps in hours (only for mode > 0)
$mode$ = 0 means constant buffering, 1 means variable buffering (extrapolation), 2 means hybrid

Output: a 2 dimensional array with $maxZones$ rows, each row contains the ordered vertices of the respective danger zone

1: <pre-processing and initialization (see **Algorithm 13**)>
2: **if** $mode == 0$ **then**
3: **for** $i = 0 \rightarrow n - 1$ **do**
4: **for** $step = maxZones - 1 \rightarrow 0$ **do**
5: $offsetV[step][2 * i] = hullV[i] + (step + 1) * bufDist * normals[i]$
6: $offsetV[step][2 * i + 1] = hullV[(i + 1)\%n] + (step + 1) * bufDist * normals[i]$
7: **end for**
8: **end for**
9: **for** $i = 0 \rightarrow n - 1$ **do**
10: **for** $step = maxZones - 1 \rightarrow 0$ **do**
11: $result[step][(i + 1)\%n] = intersectLL($
$offsetV[step][2 * i], offsetV[step][(2 * i + 1)\%(2 * n)],$
$offsetV[step][(2 * i + 2)\%(2 * n)], offsetV[step][(2 * i + 3)\%(2 * n)])$
12: **end for**
13: **end for**
14: **else**
15: **for** $i = 0 \rightarrow n - 1$ **do**
16: $dir = (hullV[i] - computeAnchor(hullV[i])).normalize$
17: $nearest = NearestPointAlgorithm(hullV[i], PerimeterXHoursAgo(fire, h))$
18: $v = \frac{orthodromicDist(nearest, hullV[i])}{h}$
19: **for** $step = maxZones - 1 \rightarrow 0$ **do**
20: $result[step][i] = hullV[i] + (step + 1) * extraHrs * v * dir$
21: **if** $mode == 2$
and $dist(result[step][i], fire.currentPerimeter.convexHull)$
$< bufDist$ **then**
22: $result[step][i] =$ < const. offset $hullV[i]$ by $bufDist$ like in $mode$ 0 >
23: **end if**
24: **end for**
25: **end for**
26: **end if**
27: **return** $result$

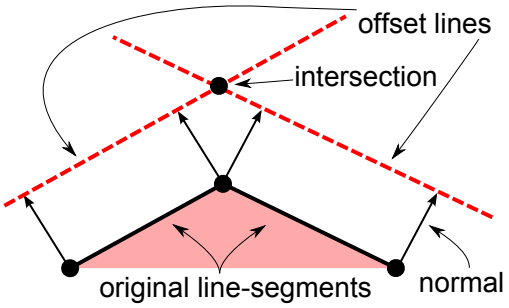

Figure 4.10: Sketch of the *Constant Outwards Offsetting*

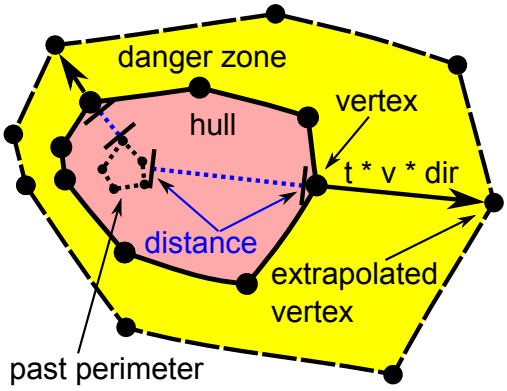

Figure 4.11: Sketch of the Extrapolation

4.8 Street Algorithm

4.8.1 Motivation

The last visualization approach targets the user-group of traffic participants. Since large bushfires can have a huge impact on the infrastructure and the traffic in the affected regions, people driving in their car to escape the fire or to reach a destination on the other side of the burns are confronted with special questions like e.g. *"How endangered are the streets in the user's environment ?"*, *"Which streets are leading into the bushfire and should thus by no means be used ?"* and *"What is the shortest evacuation route bringing the user into a not acutely endangered zone ?"*.
This visualization approach is designed to answer all of these three questions on a single map. It intuitively delivers information about how threatened specific roads in the user's environment are, by coloring the surrounding streets according to their threat level. Additionally it highlights streets leading into the fire by placing stop-signs on them and eventually, the shortest evacuation route leading the user out of the endangered zones is computed and displayed on the map.
Beside the street-data ($streets$), the input for this approach are the user GPS - position (u), the current perimeter of the fire ($peri$), the search-range ($maxDist$), a so called threat-value function ($threat()$) and a function that specifies the coloring of different threats ($getColor()$).

4.8.2 Algorithm

The algorithm itself is a variant of the *Dijkstra* [17] graph search algorithm which is a prominent greedy shortest path algorithm. It starts at the way-point in the street network that is closest to the user's GPS position and searches the environment until all reachable way-points in the given search range are visited or

located in the fire area.
During search, the algorithm constantly visits way-points by always going one step further along the shortest not yet visited path.
At each of these way-points, it computes a threat value using the function $threat$, which can be passed as a parameter to the algorithm. A threat value is a numerical representation of how threatened a given position is, with respect to the fire hazard. The design of such a threat value function can be very complex and should be done in cooperation with experts. In the prototype implementation, the threat value is approximated by the distance a point is away from the next fire or by the expected time, a fire needs to progress until the position is reached (based on a simple extrapolation similar to the *Danger Zone Algorithm*'s). This threat value is then passed to a second function $getColor$ mapping the threat value to a color that is hence used to color the just visited road-segment. Doing this at each visited way-point in the surrounding street network, all streets are finally colored according to their endangerment.
In addition to the threat value check at each way-point, an inside-test between the way-point and the current fire perimeter (or its hull) is conducted. In this way, the algorithm finds out all road-segments that lead into the fire area. These are then marked by stop-signs. Moreover, the search stops at road-segments leading into the fire which is important to the evacuation-route computation.
To compute the shortest evacuation route starting at the user's position leading into a zone rated "safe", *Dijkstra*'s shortest path property is crucial. The prototype implementation and the pseudocode beneath implement the algorithm using a priority-queue containing search nodes [17] that are sorted by ascending way-costs. Because the algorithm always pops the first element from the queue, it is guaranteed that the search always visits the next cheapest path first and this implies, that when the algorithm expands a search-node, it found a cheapest path from the user to its way-point. This enables the algorithm to find a shortest evacuation way during the search coloring the streets and detecting the road-segments leading into the fire: In case the threat value of the user's position is initially rated "endangered", the search's path from the user to the first expanded search node whose way-point is rated "safe" is a shortest evacuation route. It can be reconstructed by simply back-chaining over this search node's parent-pointers. This reconstructed path is finally colored blue on the map. An additional guarantee is, that the reconstructed evacuation route will never lead through the fire, because the search does not generate the children of search nodes in the fire and hence, a reconstructed shortest path using the parent-pointers of a "safe" way-point cannot contain a street leading through the fire.

[17] A search node is a triple containing the actual way-point in the street network, a pointer to the node's parent search node that generated the node and the total path costs from the root node (at the user-position) to the node. The root node has a NULL-parent-pointer.

4.8.3 Pseudocode

Algorithm 15 shows the pseudocode of the *Street Algorithm*.

4.8.4 Comments

This visualization method answers multiple questions at once. To maintain simplicity and intuitiveness, each question is addressed with an intuitive visualization approach:

- the street threat is communicated by directly coloring the road-segments (with color-interpolations between red (highly endangered) and green (safe)),
- streets that should not be used are marked with an internationally understandable sign: the stop-sign, that, in conjunction with the surrounding red streets, communicates "don't use this road anymore" to the user and
- the evacuation way colored in blue is a clearly noticeable contrast to the threat-colored streets that mainly use warm colors and to the uncolored normal streets outside the search range.

Implemented in a website or app, it would probably be a good approach to let the user enable and disable features like the colored streets or the stop-signs, so that he is able to focus on what he is especially interested in.

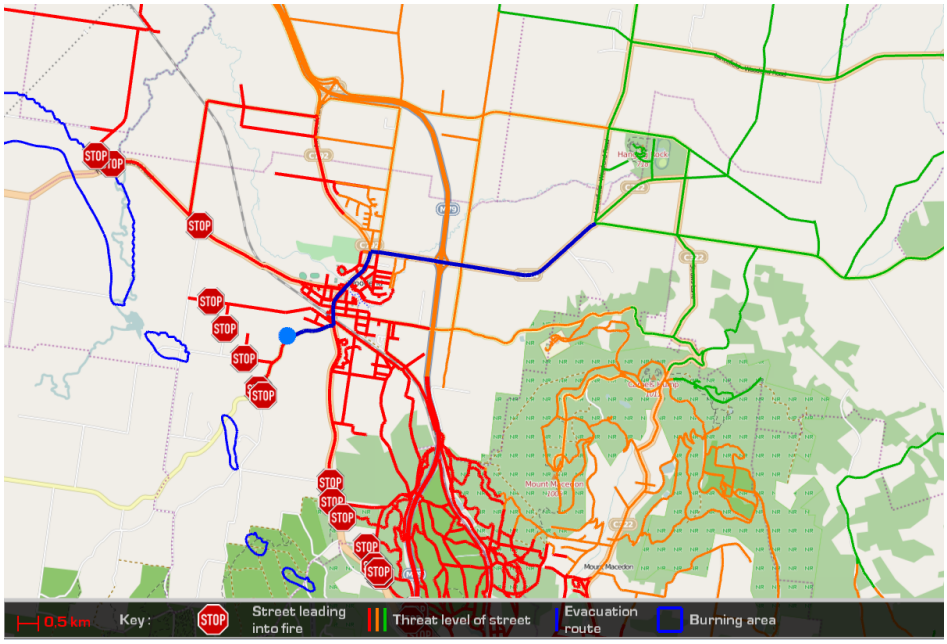

Figure 4.12: Example - Street Visualization

Algorithm 15 Street Algorithm

Input: $peri$ = the current perimeter of the fire
u = the user's GPS-position
$maxDist$ = the search range
$streets$ = the street network data
$threat()$ = a function computing a threat value for a given position
$getColor()$ = a function mapping threat values to colors
Output: the streets around the user are colored according to their threat level
road-segments leading into the fire are marked by stop-signs
the shortest evacuation route is colored blue

1: $startNode =$ new SearchNode($streets.getNearestWayPoint(u)$,$NULL$,0)
2: $startThreat = threat(startNode.location)$
3: $evac =$ false, $done =$ empty list,$todo =$ empty priority-queue
4: $todo.add(startNode)$
5: **while** $todo.size \neq 0$ **do**
6: $curNode = todo.pop()$
7: $done.add(curNode.location)$
8: **if** $startThreat \neq$ "safe" **and not** $evac$ **and** $threat(curNode.location) ==$ "safe" **then**
9: $evac = true$
10: < color evacuation way ($curNode.getPathToRoot()$) blue >
11: **end if**
12: **for all** WayPoints $neighbor$ adjacent to $curNode.location$ in $streets$ **do**
13: $neighborNode =$ new SearchNode($neighbor$, $curNode$, $curNode.cost + cost($ road segment from $curNode.location$ to $neighbor$))
14: **if** road segment from $curNode.location$ to $neighbor$ not colored **then**
15: < color road segment from $curNode.location$ to $neighbor$ with $getColor(threat(neighbor))$ >
16: **end if**
17: **if** $inside(neighbor, peri)$ **then**
18: < add stop-sign to road segment from $curNode.location$ to $neighbor$ >
19: **else if** $neighbor \notin done$ **and** $dist(neighbor, u) < maxDist$ **then**
20: **if** $neighbor$ already in some SearchNode $beforeNode$ in $todo$ with $beforeNode.cost > neighborNode.cost$ **then**
21: $todo.remove(beforeNode), todo.add(neighborNode)$
22: **else if** $neighbor$ not already in some SearchNode in $todo$ **then**
23: $todo.add(neighborNode)$
24: **end if**
25: **end if**
26: **end for**
27: **end while**
28: **return**

4.9 Conclusion

Seven different approaches to visualize bushfire incidents were introduced in this chapter. Besides the raw fire-perimeter data, most of them are based on additional input parameters (as can be seen in the overview table Figure 4.1) that can be assumed to be available in mobile apps and websites. They all use these parameters in the transformation stage to produce visualizations that provide more information than current visualizations. For this, the transformations used range from simple geometric calculations as in the *Nearest Point Algorithm* to more complex analyses as in the *Danger Zone Algorithm* or the *Street Algorithm*. All visualizations are designed to answer particular questions that casual users are confronted with in bushfire regions. To answer these questions, the relevant information is visualized on the map.
In the visualization stage, a variety of different 2D-visualization techniques is applied. They range from simple marker placements over lines, arrows, curves to polygon - areas and they use visual variables like color and transparency to emphasize temporal development and uncertainty. Beside the static approaches for spatio-temporal visualization, two animated visualization techniques are proposed.

One last visualization feature will be introduced now. It is a simple way to provide some general information about the threat at the user's location and it can be used in conjunction with all introduced visualizations:
Besides the usage in the *Street Algorithm*, the concept of a threat value representing the threat at a given position originating from a bushfire can be used to adapt the visualization of the user-marker on the map. In an application where the user's position is marked on the map, the marker can be animated to pulsate with increasing frequency when the threat value at the user's position increases and it could change color according to a $getColor$ function. The advantage is, that this feature can be combined with all visualization techniques showing the user on the map once an appropriate threat value function is designed. It is also implemented in the prototype system (see Figure 4.13) and used in conjunction with all introduced visualization techniques.

Figure 4.13: Example - Threat Value User-Marker

Chapter 5
Implementation

This chapter will give an overview over the prototype implementation developed for this thesis. It will briefly discuss the software architecture, outline the most important features and introduce the Graphical User Interface.

5.1 Overview

After developing new concepts for the visualization of bushfires, the next goal was to implement them in a prototype system. Therefore, a Java program was developed that implements all introduced algorithms in Java as well as all visualizations that are currently used in public warn-applications. The program's main tasks are to:

- test and demonstrate the new visualizations
- compare them with the current ones
- generate visualizations suitable for the use in the evaluation - phase of this thesis
- enable finding good parameter configurations for all algorithms by providing a possibility to easily change parameters

5.2 Software Architecture

The following diagram illustrates the program's main workflow that is closely related to the classical visualization pipeline introduced in the **Related Work** (3.1) section. As can be seen in the diagram, the three main stages *Acquisition*, *Transformation* and *Visualization* are represented in the software architecture too.

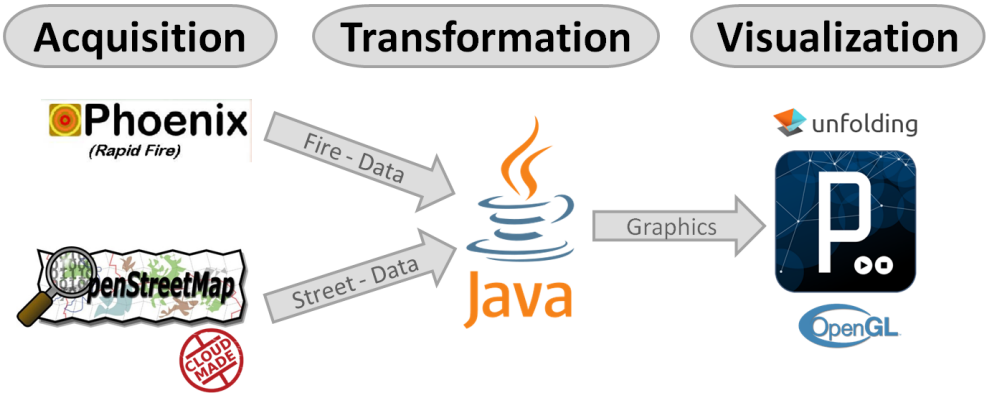

Figure 5.1: The Software Architecture Pipeline

While the *Acquisition* of course is not directly part of the software, their results are fed into the program. The most important inputs from the *Acquisition* stage are the fire-data and the street-data.
The fire-data used in the prototype system was provided by the *University of Melbourne* and consists of shapefiles[18] containing the fire-perimeter and surface data of 5 bushfires located in the Melbourne area that have catastrophic extents. These shapefiles contain chronologically ordered layers with geo-referenced polygons depicting the fire perimeters at 5 moments in time each with a distance of 1 hour. The files originate from runs of the expert bushfire simulation software *Phoenix Rapid Fire* [18] and are read by the developed fire-loading methods which use an open-source shapereader-library by Thomas Diewald [19] to access the shapefile's content. After loading the shapefiles, the fires are translated to the internal classes and added to the internal model. This model manages the software world-state. It has a list of all loaded fires and information about the user and the map as well as the street data. The internal classes are modeled similar to the fire - geometry data introduced in section 4.1. A fire object for example contains an point of ignition and a list of perimeter-objects, which again contain lists of polygons and so on. Consequently, a set of classes provides data-structures to model all important entities that occur in the bushfire setting and the internal model representing the current bushfire scenario is maintained by the program.
The second important input, the street-data used in the implementation of the

[18] `https://en.wikipedia.org/wiki/Shapefile` [last accessed Sept. 5, 2013]
[19] `https://github.com/diwi/diewald_shapeFileReader` [last accessed Sept. 5, 2013]

Street Algorithm, is provided by *CloudMade*[20]. Concretely, a database file with *OpenStreetMap*[21] - data of streets in the fires' environment (here: Victoria in Australia) is used by the program which can then run the *Street Algorithm* on the street network contained in these files. To access the *OpenStreetMap* files, some functions of the open-source Java library *Traveling Salesman*[22] by Marcus Wolschon and the library's fast binary-database format for street-map data are used.

After loading all necessary data and establishing the current bushfire scenario in the internal model, the *Transformation* stage takes over. This stage is completely written in Java. It encompasses the core algorithms for the visualizations introduced in **Chapter 4**, additional algorithms used, like for example the *Graham-Scan-Algorithm* computing the convex hull or ray-casting algorithms for intersection checks, and everything else that is related to the program's management of the internal model and computations. To improve performance and to remain responsive, many algorithms and especially all visualization algorithms introduced are programmed to run concurrently. In case they need some time to finish (like for example in an extensive street search), all intermediate results (like colored streets or already found stop-signs) are displayed as soon as the algorithm computed them.

The last stage finally manages the graphical output, which includes the rendered map and the actual visualizations. For this, the *Visualization* stage is implemented using the open-source visualization framework *Processing*[23] for Java which uses *OpenGL*[24] to improve performance. For the geo-visual output, a slippy map is implemented and rendered using the support of the *Unfolding Maps*[25] extension by Till Nagel for *Processing*.
Additionally, the program offers a GUI which will be introduced in the section **Features and Graphical User Interface** (5.3). It is implemented using *Java Swing*[26] in conjunction with the *SeaGlass Look and Feel*[27] and it consists of four main sections: a control panel to manage the scenario, a panel to configure the algorithms' parameters, a menu-bar and an area where the slippy map containing all results and the key are rendered on the screen.

Because of the program's modular architecture, extending the model and the

[20] `http://downloads.cloudmade.com/oceania/australia_and_new_zealand/australia/victoria#downloads_breadcrumbs` [last accessed Sept. 5, 2013]
[21] `http://www.openstreetmap.de/` [last accessed Sept. 5, 2013]
[22] `http://sourceforge.net/projects/travelingsales/` [last accessed Sept. 5, 2013]
[23] `http://processing.org/` [last accessed Sept. 5, 2013]
[24] `http://www.opengl.org/` [last accessed Sept. 5, 2013]
[25] `http://unfoldingmaps.org/` [last accessed Sept. 5, 2013]
[26] `http://de.wikipedia.org/wiki/Swing_%28Java%29` [last accessed Sept. 5, 2013]
[27] `http://seaglass.googlecode.com/svn/doc/index.html` [last accessed Sept. 5, 2013]

GUI is uncomplicated. It exists a class which encapsulates all visualization-algorithms and new ones can be added straightforwardly. In addition, settings for the window (e.g. resolution of map, resolution of window), the graphics quality (e.g. Anti-Aliasing) and other properties are saved in and loaded from a config-file, which allows to configure the program for different machines in a way that matches the hardware.

5.3 Features and Graphical User Interface

As already mentioned, the prototype system's GUI consists of four main sectors, which are identified in Figure 5.2:

A *slippy map*[28] (Sector 1) that displays the current scenario with its fires, the user's position and of course the results of the visualization algorithms applied as well as an appropriate key. It uses the map service providers *OpenStreetMap* for topographic road maps and *Microsoft Aerial* for hybrid satellite images with roads. Switching between satellite and road-map mode is possible by simply pressing the corresponding button on the right side of the window.

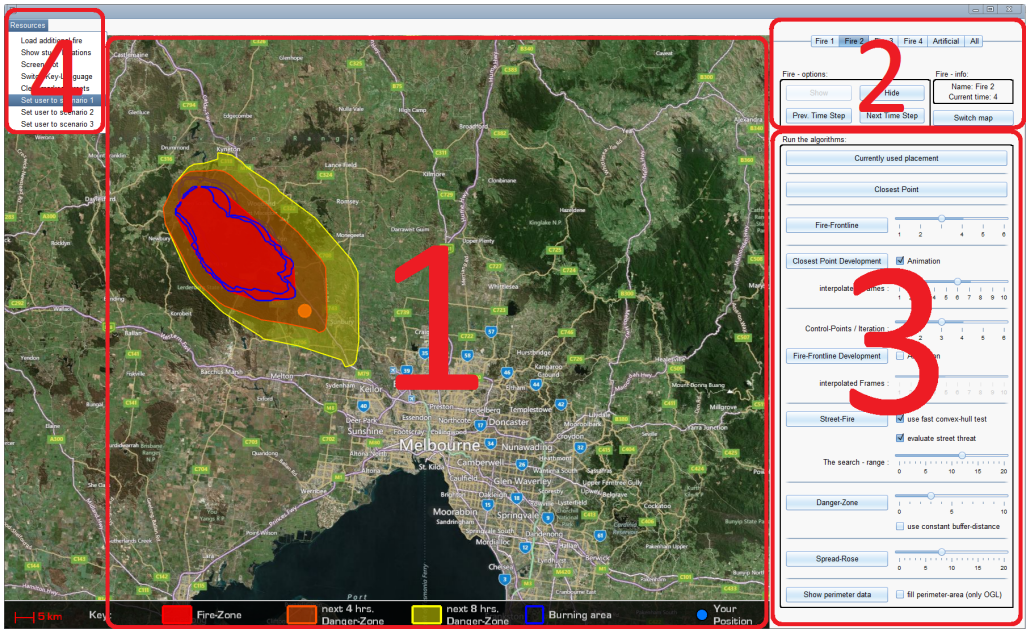

Figure 5.2: The Main Sectors of the Prototype Program's GUI

Besides this, the program enables the user to configure the scenario that is visualized with the corresponding control panel (Sector 2) and the map itself. One

[28] a map that can be zoomed and panned, as known from websites like *OpenStreetMap* or *Google Maps*

can for example move the user in the scenario by clicking at a position on the map, select the fires one is interested in, enable or disable the rendering of a perimeter and one can let the program show the convex hull of a fire. Moreover it is possible to go back and forth in time to see the fire's development over time and to select the moment in time that the algorithms should be applied to.

As one of the program's main goals is to test, demonstrate and configure the algorithms, the algorithm-panel (Sector 3) offers elements like sliders, checkboxes and buttons to allow the user to configure, set and change the crucial parameters for each algorithm. In the following, he can apply them to the configured scenario by pressing the respective algorithm-button.

Since the visualizations that are used in the evaluation phase should be generated with the prototype program, a few more features like loading additional fires at runtime, loading whole predefined scenarios and the functionality of taking screenshots of the map extract are integrated in the menu-bar (Sector 4) in the upper left corner of the window.
At any time, an appropriate key for the currently applied algorithm is shown in the slippy map, the map's scale is illustrated and it is possible to change the key-language (German to English or vice versa) by pressing the corresponding menu-bar button. For finally taking a screenshot of the whole visualization displayed on the map, the user can save the current extract together with the key as a *.png* image by pressing the "Screenshot" button in the menu-bar.

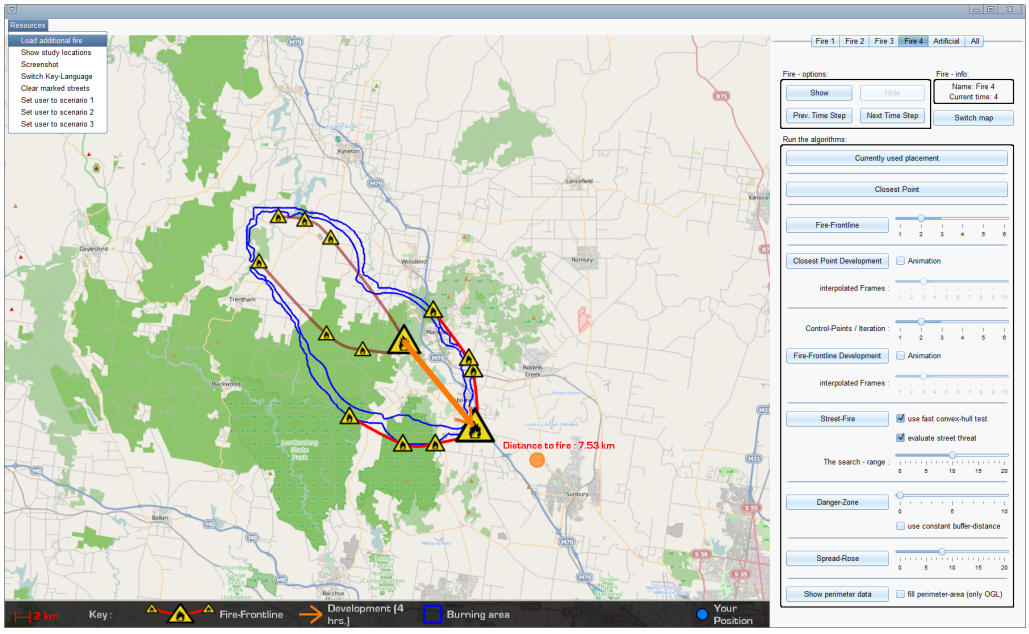

Figure 5.3: Screenshot of the Prototype Program

Chapter 6
Evaluation

This chapter concerns the last goal of this thesis: the evaluation of the developed visualization approaches.
For this, the conducted study's structure is outlined and the study's most interesting results are described and discussed.

6.1 Overview

After developing new visualization approaches for bushfires and after developing a prototype system that implements all approaches, the last step is to evaluate the visualizations in order to find out, which visualizations suit best for the application in public bushfire warn apps and websites. The main goals of the study are thus to

- assess the communicated threat of the different approaches in several scenarios
- compare all approaches with respect to the communicated threat
- assess the visualizations' popularity amongst users
- gather some general information about the users and their usage of maps

To achieve these goals, several types of evaluation are suitable. One possibility is to invite several participants and to let them use the prototype program in order to explore different scenarios and to rate their threat. Another possibility is to invite participants in order to let them fill in a questionnaire, showing examples of visualizations. But both lab-study approaches have one disadvantage in common: the number of participants is limited due to the time-consuming supervision.

In order to assess data from many participants of the right user-group (users of public apps and websites that are laypersons in the field of fires), the evaluation in this thesis' context is designed as an online-study. The advantage here is that the study is freely and at every time available for everybody willing to participate. Moreover the web-approach is close to the real application environment, the visualizations are designed for, since they are shown in a web-browser on a PC or mobile device.

6.2 Study Structure

The online-study is implemented using the web-survey system *LimeSurvey*[29] and available in English and German. With this system it is possible to display screenshots showing visualizations generated with the prototype program to the participant in order to ask him questions about what he sees. The survey is constructed as follows:

Scenarios

The survey's main task is to assess the threat communicated by different visualizations of bushfires in order to enable a following comparison of the approaches. To achieve this goal, three different scenarios were set up, each consisting of a specific fire at a specific location and a user position:

- **Scenario 1**: A fire with a great distance to the user (15 km), a less dangerous situation
- **Scenario 2**: A fire with mid-range distance to the user (6 km), a dangerous situation
- **Scenario 3**: A fire close to the user (3 km), a highly dangerous situation

For each of these three scenarios, ten different visualizations were generated using the prototype implementation:

1. the currently used simple *Perimeter Visualization*
2. the currently used *Ignition Marker Placement*
3. the *Nearest Point Visualization*
4. the *Nearest Point Development Visualization* (static)
5. the *Fire Frontline Visualization*
6. the *Fire Frontline Development Visualization* (static)

[29] `http://www.limesurvey.com/` [last accessed Sept. 5, 2013]

7. the *Danger Zone Visualization* with a 4 and an 8 hour danger zone
8. the *Spread Visualization* with 8 arrows
9. the *Street Visualization* with colored streets
10. the *Street Visualization* without colored streets

This results in a total of $3 * 10 = 30$ different map extracts presented to the user.

Questions

But before the user rates the threat of the scenarios using the 30 map-extracts, a first calibration question is asked, measuring the participants anxiety or cautiousness. In this first question, a map extract showing the perimeter of an artificial circular fire and 10 positions adjacent to each other (with a distance of 2 km between two subsequent positions) is given. The user is then asked

Please estimate, up to which position you would rate the threat emanating from the fire to be extremely high.
After that please estimate, from which position on you would rate the threat emanating from the fire to be very low.

To answer, the participant is supposed to click on the radio-buttons that belong to the chosen positions. This introductory question is designed to check if a consensus about the threat originating from a fire exists amongst all participants.

After this first question, the study's main part begins. An overview of all visualizations with small examples is shown to the participant explaining what the particular visualizations display and that all of them will appear during the next 30 questions. In these 30 following questions, all 30 map extracts are shown (one at a time) in a previously defined but random order, together with the question

How would you rate the threat emanating from the fire at the location marked by the blue dot ?

The user's task is then to look at the visualization shown and to rate how threatened the marked user position (blue dot) is on a discrete scale of 0 to 5, where 0 means *no threat at all* and 5 means *acute danger of life*. All intermediate steps are interpolations between these two extremes.
Finally, these questions assess the visualizations' communicated threat, because the participants are not informed that they will only be confronted with three different scenarios, each scenario 10 times. To ensure that the participants are not misinterpreting distances, the map extract shows, beside the visualization's key, the map's scale which remains fixed in all questions.

After completion of all scenario visualizations, the user's favorite visualizations are assessed: The first popularity question

Which visualizations give the best information to you ?

shows an overview of all 10 visualizations that were shown in the scenarios before and allows the participant to mark up to three favorite visualizations. Similarly, the two following popularity questions assess whether there is a difference between favorites for websites and for apps by asking

Thinking of a mobile Bushfire-warning-application (e.g. for a smartphone), which visualizations would you prefer ?

and

Thinking of a Bushfire-warning website, which visualizations would you prefer ?

Finally, some general information about the participant is gathered: The question

How often do you use electronic maps (e.g. Google Maps, mobile navigation systems, etc.)?

and

How often do you use paper-maps (e.g. road maps, atlases, etc.)?

with answer-possibilities *everyday, every week, every month, a few times per year* and *hardly ever / never* provide information about the participants usage of maps and the question

How often do you use OpenStreetMap ?

with the same possible answers, checks whether a participant is used to the map extracts' map-style.
The very last questions ask for the participant's gender and age and ultimately, a possibility to leave text comments is given.

Overall, it took the participants 16 minutes in average to complete all these questions.

6.3 Study Results

This section will present the most interesting results of the study. The focus lies on an overview over the gathered data, the comparison of the visualization approaches, the popularity results and noticeable trends. Significance tests and correlation tests are beyond the scope of this bachelor thesis and thus not part of this evaluation.

General

The first interesting results are some general statistics related to the participants and their usage of maps:
At the time of evaluation, 107 participants completed all questions, with their age ranging from 16 to the age of 66. The average age of all participants was 32 years with a standard deviation of 15.4 years. Furthermore, 35 participants (32.71%) were female and 72 (67.29%) male.

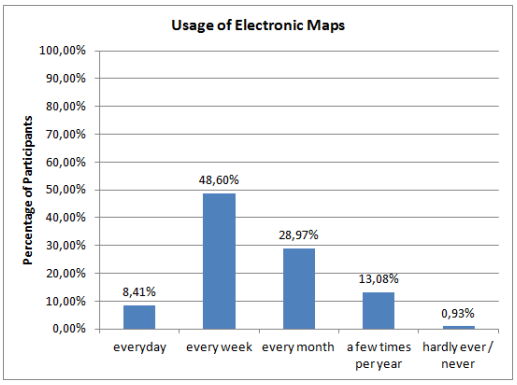

Figure 6.1: The Usage of Electronic Maps

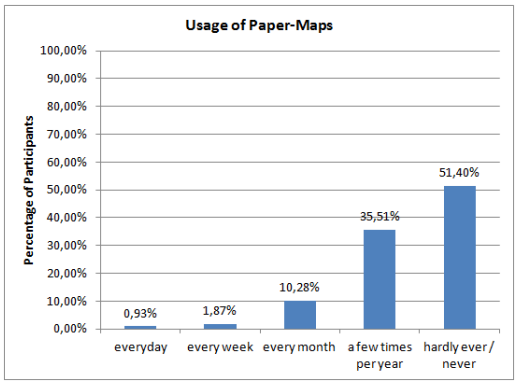

Figure 6.2: The Usage of Paper-Maps

The results of the questions about the usage of maps indicated, that the set of participants was a good representation of the targeted user group, which were users of public warn apps and websites. While only a few participants were used to the displayed map-style of *OpenStreetMap* (more than 82% used *OpenStreetMap* at most a few times per year, 66 % hardly ever or did not know it), Figure 6.1 shows that the general usage of electronic and online maps was wide-spread amongst all participants since more than 85% of them used electronic maps at least once a month. On the other side, the participants rarely used non-electronic paper-maps. As Figure 6.2 shows: only about 13% of them use paper-maps on a monthly basis.
This means, the participants could mainly focus on the displayed visualizations and did not have to break new ground using the online map-extracts, since online maps were, in principle, a familiar environment for them.

The next part of the evaluation addresses the communicated threat. The evaluation of the introductory calibration question found, that the majority of the participants generally agreed on estimating a bushfire's threat. 78 % estimated the threat originating from a fire to be very high up to a distance of at most 6 km. At the same time, 70 % rated a bushfire's threat to be negligible from a distance of 12 km or more on. These results mostly match the participants' estimates in the following scenario questions, since here, the scenario with a distance of 3 km was most widely recognized as very dangerous, the scenario with 6 km distance was rated generally dangerous and the scenario, where the user is 15 km away

from the fire, was mostly rated moderately or less dangerous.

Comparison of the Communicated Threat

The main part of the scenario-evaluation is the comparison of all different visualization approaches with respect to the communicated threat. The results are summarized in the following table showing the visualizations ranked by their averagely communicated threat and the diagrams in Figure 6.3, 6.4 and 6.5 depicting a graphical comparison:

Statistical Abbreviations:
$\tilde{x}$ = median of communicated threat; $\emptyset_x$ = average of communicated threat;
σ_x = standard deviation of communicated threat
Reminder:
threat is measured on a discrete scale of 0 (*no threat at all*) to 5 (*acute danger of life*)

Abbreviations for the Visualizations:

P	simple Perimeter	**FFD**	Fire Frontline Development
IM	Ignition Marker	**SP**	Spread
NP	Nearest Point	**DZ**	Danger Zone
ND	Nearest Point Development	**S (c)**	Street (colored)
FF	Fire Frontline	**S (nc)**	Street (not colored)

Table 6.1: Abbreviations for the Visualizations

Scenario 1 (15 km)				**Scenario 2** (6 km)				**Scenario 3** (3 km)			
Vis.	$\tilde{x}$	$\emptyset_x$	σ_x	Vis.	$\tilde{x}$	$\emptyset_x$	σ_x	Vis.	$\tilde{x}$	$\emptyset_x$	σ_x
DZ	3	2.98	1.14	**DZ**	4	4.30	0.79	**DZ**	5	4.58	0.81
FFD	3	2.76	1.16	**FFD**	4	3.65	0.90	**S (c)**	5	4.48	1.02
SP	3	2.55	1.12	**ND**	4	3.57	0.86	**FF**	5	4.48	0.81
ND	2	2.45	1.12	**S (c)**	4	3.47	1.02	**FFD**	5	4.42	0.78
P	2	2.39	1.10	**SP**	3	3.16	0.93	**SP**	4	4.33	0.83
S (nc)	2	2.16	1.18	**S (nc)**	3	2.97	0.95	**NP**	4	4.27	0.85
FF	2	2.13	1.10	**FF**	3	2.91	1.00	**ND**	4	4.13	0.94
S (c)	2	1.92	1.09	**P**	3	2.88	1.07	**S (nc)**	4	3.90	1.15
NP	2	1.84	1.16	**NP**	3	2.73	1.03	**P**	3	3.30	1.05
IM	0	0.60	0.96	**IM**	1	1.46	1.12	**IM**	2	2.38	1.02

Table 6.2: Comparison and Ranking of the Communicated Threats

Analyzing the table, it is remarkable that the *Danger Zone* (**DZ**) visualization warned strongest in all scenarios. Similarly, the spatio-temporal visualizations involving comparably much transformation and analysis of the data, especially the *Fire Frontline Development* (**FFD**), warned in most cases strongly and the *Spread*

visualization (**SP**) remained mid-table through all scenarios..
Comparing the new visualizations with the currently used ones, it is noteworthy that the *Perimeter* (**P**) visualization communicated in all scenarios less endangerment than the *Danger Zone*, the *Fire Frontline Development*, the *Spread* visualization and the *Nearest Point Development* (**ND**) approach, which are spatio-temporal. In scenario 2 and 3, even the colored *Street* (**S (c)**) visualization threatens the users more than the *Perimeter* depiction .
Moreover, the *Perimeter*'s communicated threat decreases relative to the new visualizations' when the user comes closer to the fire and the situation becomes increasingly dangerous. This can be seen by looking at the *Perimeter* ranking in the scenarios. In scenario 1, the Perimeter is ranked 5th, in the following scenario the ranking drops to 8th and in the most dangerous scenario, the *Perimeter* is ranked second to last above the *Ignition Marker* (**IM**), with a median value of 3 while the *Danger Zone* records a median of 5. This shows, that the closer the fire is, the less people are warned using the *Perimeter*, in comparison to the new visualizations.
The *Ignition Marker* itself is ranked last in all scenarios.

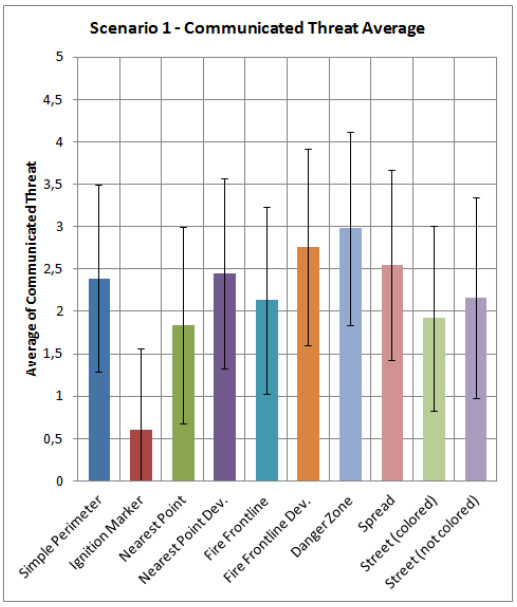

Figure 6.3: Comparison of the Communicated Threat in Scenario 1 (15 km)

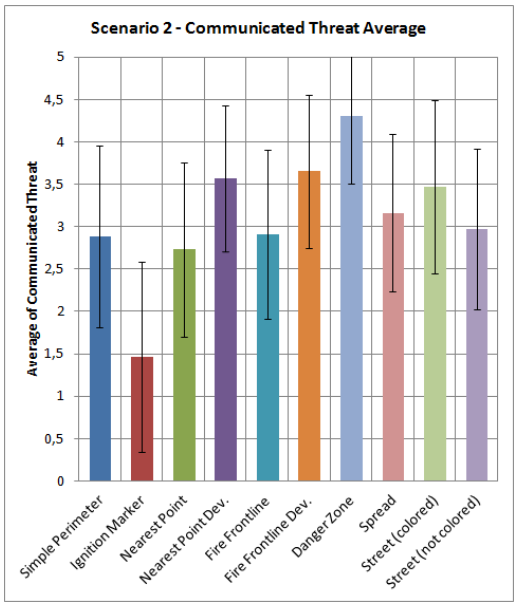

Figure 6.4: Comparison of the Communicated Threat in Scenario 2 (6 km)

Especially interesting is the difference between the first and last visualization, the *Danger Zone* and the *Ignition Marker*: In each scenario, their communicated threat differs by 3 median values, which is huge when considering a scale of 0 to 5. As an example, the threat communicated in scenario 3, the most dangerous scenario where the user is only 3 km away from the fire, the *Danger Zone* communicated this acute danger with an average value of 4.58 and a maximum median value

of 5 whereas the *Ignition Marker* still communicated a median value of 2 with an average of 2.38, which is even less than the *Danger Zone*'s value for scenario 1. Thus, the currently most wide-spread visualization method communicates by far the least threat, compared to all introduced alternatives.
A further detail is the behavior of the colored *Street* visualization. In scenario 1, the user's position was located next to streets rated "safe" by the threat value function and thus colored green. In the second scenario, the user's environment was colored orange and in the last scenario of course, the streets around the user were colored red. Analogue to the streets' color, the communicated threat as well as the ranking relative to all other visualizations increased from median value 2 and rank 8 in scenario 1 over median value 4 and rank 4 in scenario 2 to the maximum median value 5 and rank 2 in scenario 3. These communicated threat values might possibly be connected with the coloring and people might tend to believe the coloring disregarding distances and other indicators. If this is the case, it would emphasize that developing good threat value and coloring functions is crucial.

The diagrams 6.3 to 6.5 illustrate the comparison graphically and it is clearly noticeable, that the danger communicated by all visualizations increased, when the user's distance to the fire decreased.
With increasingly dangerous scenarios, the new concepts provided more warning compared to the currently used ones and the *Ignition Marker* always communicates least threat.
In the two first and not highly critical scenarios, the spatio-temporal visualizations *Danger Zone*, *Fire Frontline Development*, *Spread* and *Nearest Point Development* communicated a higher threat value than non-temporal visualization. In a critical situation (Figure 6.5) though, one new visualization is just as good as the other while the current visualizations both still communicate less danger.

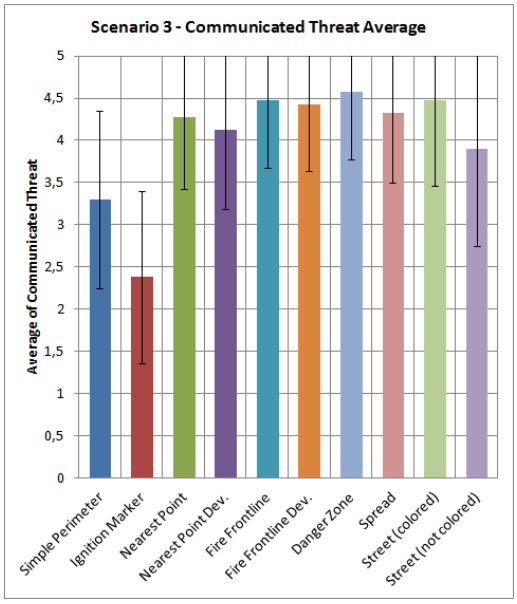

Figure 6.5: Comparison of the Communicated Threat in Scenario 3 (3 km)

Popularity

The second main part of the evaluation is the acceptance of the approaches amongst users and the popularity of the visualizations. The following table

and the diagrams 6.6, 6.7 and 6.8 summarize the results of the three popularity questions asking for the user's favorites in general, with respect to mobile and with respect to website applications. (Abbreviations see in the **Comparison of Communicated Thread** (6.3) section above - Table 6.1)

General Popularity			Mobile Popularity			Web Popularity		
Vis.	Votes	%	Vis.	Votes	%	Vis.	Votes	%
DZ	86	31.97%	**DZ**	80	32.52%	**DZ**	71	27.63%
S (c)	50	18.59%	**S (c)**	49	19.92%	**S (c)**	50	19.46%
FFD	41	15.24%	**FFD**	37	15.04%	**SP**	44	17.12%
SP	37	13.75%	**SP**	32	13.01%	**FFD**	38	14.79%
P	21	7.81%	**P**	18	7.32%	**P**	24	9.34%
ND	11	4.09%	**ND**	13	5.28%	**ND**	15	5.84%
FF	10	3.72%	**FF**	6	2.44%	**FF**	6	2.33%
S (nc)	8	2.97%	**S (nc)**	6	2.44%	**S (nc)**	5	1.95%
IM	4	1.49%	**IM**	3	1.22%	**IM**	2	0.78%
NP	1	0.37%	**NP**	2	0.81%	**NP**	2	0.78%
	269	100%		246	100%		257	100%

Table 6.3: Comparison and Ranking of the Popularity

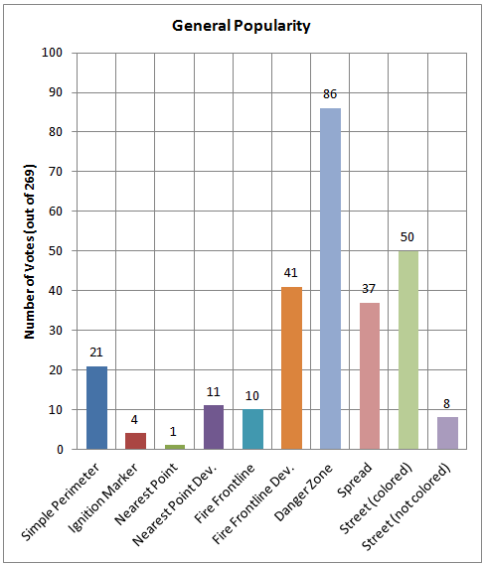

Figure 6.6: The General Popularity of all Visualizations

Obviously, 4 clear favorites exist: the *Danger Zone* approach, the colored *Street* approach, the *Fire Frontline Development* visualization as well as the *Spread* visualization. These four new approaches rank top 4 in all three popularity questions and it is noticeable, that these are the algorithms that involve most transformation computation and analysis. Three of them are spatio-temporal visualizations and the fourth is the *Street* algorithm.

A further remarkable connection is to the results of the communicated threat: the four most popular approaches are also the approaches, that communicated a comparably high threat and that, in most cases, communicated a threat higher than the *Perimeter* depiction (and consequently higher than the *Ignition Marker*), whose popularity lags far behind the four favorites'.

All other visualizations, namely the *Nearest Point*, the *Fire Frontline*, the *uncolored*

Street visualization, the *Nearest Point Development* approach and the wide-spread *Ignition Marker* are unpopular and gathered in each question less then 13% of the votes in total together. One factor that certainly plays a role here is, that the *Fire Frontline Development* approach comprises the *Nearest Point*, the *Nearest Point Development* approach and the *Fire Frontline* approach in some way. It also shows where the nearest point is, it shows the current frontline and it shows the development thereof. Similar, the *Spread* algorithm comprises a perimeter depiction by augmenting it with spatio-temporal information about the spread behavior and the colored *Street* approach is an extension of the uncolored version.

All these results show, that the users want visualizations other than the currently used ones and that the introduced new visualization approaches are suitable alternatives. The results also imply, that users prefer visualizations that warn comparably strong and that involve some kind of analysis and transformation to provide more information about the situation and the bushfire than offered by current visualizations.

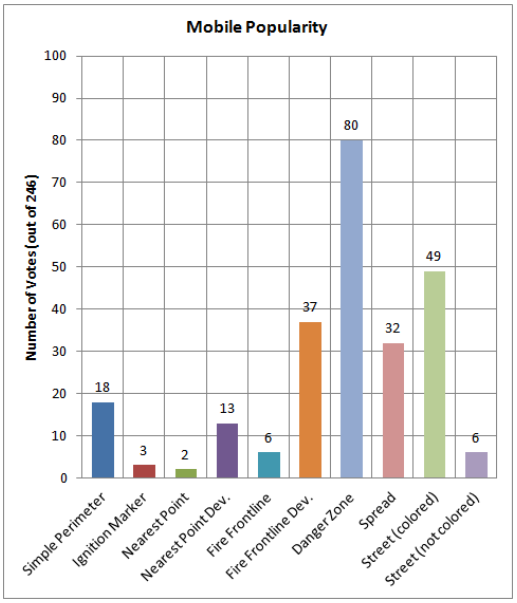

Figure 6.7: The Mobile Popularity of all Visualizations

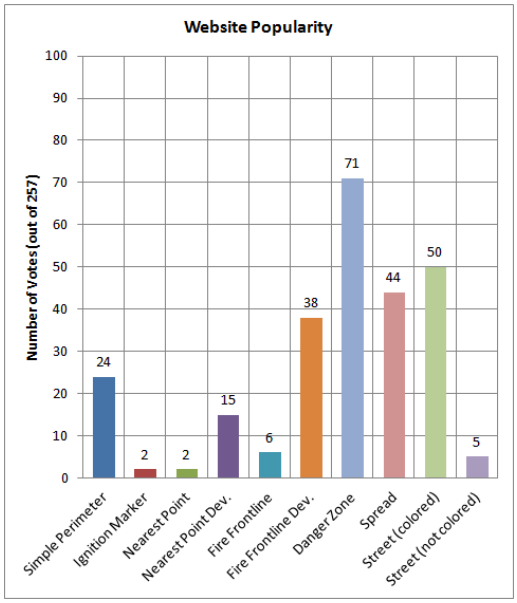

Figure 6.8: The Web Popularity of all Visualizations

As a last result, the comparison of Figure 6.7 and 6.8 shows, that there is no noteworthy difference between the desired visualizations in mobile apps and websites. Except that the *Spread* visualization and the *Fire Frontline Development* swapped places in the ranking of the website popularity, the participants' favorites remain the same.

Chapter 7
Conclusion and Outlook

This chapter concludes the thesis and provides some recommendations for further work in the area.

7.1 Conclusion

The goal of this work was to develop several new approaches to visualize bushfire incidents in public warn apps and websites. Additionally, these new concepts were to implement in a prototype system and should be evaluated in order to compare them with the current approaches, for finding out whether they are suitable alternatives for the visualization in public warn applications.

In the related literature there are some important concepts like the *Visualization - Pipeline*, the *completeness of data vs. comprehensibility of data tradeoff*, the concept of *visualizing uncertainty* and the concept of *"Visualizations answering particular questions"*. All these concepts were taken into account when developing the new visualization methods.
The currently available public warn apps and websites mainly use a simple static marker-placement at the fire's location of ignition and some recent websites additionally added a depiction of the most up-to-date perimeter. But both approaches are static, do not include the temporal dimension and are not personalized in a way that the visualization adapts to the user's situation. Moreover, no additionally available information like data about streets in the environment or the user's position is used for the bushfire's visualization.
Besides these public visualizations, expert systems like bushfire prediction and simulation programs visualize bushfire data in a professional and scientific way. These visualizations contain very much information and are very accurate, but

usually, the knowledge of an fire-expert or a scientist is needed in order to decode the information in the visualization and to understand what is visualized and what it means. Consequently, these expert visualizations are not suitable for laypersons.

For filling the gap between complex expert visualizations and less informative public ones, the 7 new visualization concepts for bushfires

1. *Nearest Point*
2. *Nearest Point Development*
3. *Fire Frontline*
4. *Fire Frontline Development*
5. *Spread*
6. *Danger Zone*
7. *Street*

and the corresponding algorithms were developed, each answering different questions and each using different visualization techniques like markers, colored lines, areas, curves and animations. To additionally improve the communicated information, parameters describing the spatio-temporal development, the user's position and information about the streets are included in the algorithms and eventually used for visualization.

To demonstrate, test and compare the new visualization algorithms, a prototype system was developed in Java implementing all current and new visualizations in an environment that allows users to set up a bushfire scenario, using bushfire data from the Phoenix simulation, and that enables them to set and change all important parameters for the algorithms. The visualizations generated with this program were used in the final evaluation.

In this final part of the work, the developed approaches were evaluated and compared to each other as well as to the currently available public visualizations. An online-study with 107 participants was conducted with the main goal to assess the threat communicated by different visualizations as well as the visualizations' popularity amongst users.
The results showed that the threat communicated by the new approaches, especially the spatio-temporal visualizations, was, in most tested scenarios, higher than the threat communicated by the currently used *Perimeter* depiction and the *Ignition Marker*. This difference was most prominent in a critical and dangerous scenario, where the current visualizations made the users underestimate the

situation.
The analysis of the popularity questions showed that the spatio-temporal approaches *Danger Zone*, *Fire Frontline Development* and *Spread* together with the *Street* visualization were most popular and the two current visualizations were considerably less popular.

These results revealed, that the users would like to see new visualizations in public bushfire warn apps and websites that involve some kind of analysis and the integration of the spatio-temporal dimension. Moreover the popularity results and the results of the communicated threat showed, that the developed approaches are suitable alternatives for public warn apps and websites, which fulfills the initial goal of the thesis.
This thesis provides an overview over the new visualization techniques, their algorithms as well as over the threat communicated by them. Developers of bushfire warn apps and websites now have to decide, in conjunction with experts, which level of danger should be communicated in their application by using suitable visualizations, in order to enable an appropriate decision-making by the users.

7.2 Recommendations for Further Work

Based on the results of this thesis, it would be interesting to investigate further animated visualization techniques of the spatio-temporal bushfire development and to compare them and their communicated threat with the approaches introduced in this work. Besides this, another aspect that could be integrated and that has also a spatio-temporal dimension is the development of smoke in the fires' environment. Suitable visualizations of the smoke would certainly improve the population's awareness of the danger originating from a bushfire.

Moreover, some concepts introduced here could be improved. For example the introduced *Danger Zone* algorithm's extrapolation: For this, it would be interesting to find out, whether there are further parameters concerning weather or fuel available online and suitable for the integration in the extrapolation approach, to approximate the future development of a bushfire more accurately. Similarly, the development of a good threat-value function in collaboration with bushfire experts would be very helpful, in order to enable algorithms using threat-value functions, like the *Street* algorithm, to rate points of interest accurately.

On the application side, the next step would be to integrate the new visualizations into public warn apps and websites, or to develop a new mobile app or website that integrates the new visualizations developed in this thesis. Of course, the application of the algorithms and visualizations is not limited to smartphone apps and websites, but generally possible for all device-classes that are able to

gather information about bushfires and other parameters, for example by using the web.
One idea would be to integrate bushfire visualizations like the *Street* algorithm in mobile navigation systems, informing the driver about the current fires on his route or in his environment. The routing algorithms could additionally be modified using a threat value function as introduced here, to compute routes that are "safe" and lead around the fires.

List of Algorithms

List of Tables

List of Figures

Bibliography

[1] WIKIPEDIA: *Bushfires in Australia — Wikipedia, The Free Encyclopedia"*. http://en.wikipedia.org/w/index.php?title=Bushfires_in_Australia&oldid=559489409. Version: 2013. – [Online; accessed 26-July-2013]

[2] WIKIPEDIA: *Wildfire — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Wildfire&oldid=564150631. Version: 2013. – [Online; accessed 26-July-2013]

[3] VICTORIA WEBSITE, The S.: *What Causes Bushfires on Public Land in Victoria?* http://www.dse.vic.gov.au/fire-and-other-emergencies/fire-management/causes-of-bushfire. Version: 2013. – [Online; accessed 26-July-2013]

[4] WIKIPEDIA: *Black Saturday bushfires — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Black_Saturday_bushfires&oldid=564980254. Version: 2013. – [Online; accessed 26-July-2013]

[5] KUNZ, Melanie ; LIENERT, Christophe ; HURNI, Lorenz: Understanding the risk: establishing of an interactive system for the visualization and exploration of natural hazards and associated uncertainties. In: *Proceedings of 24th International Cartographic Conference (ICC). Santiago de Chile*, 2009, S. 1–11

[6] BLACK, Julian ; ARROWSMITH, Colin ; BLACK, Michael ; CARTWRIGHT, William: Comparison of Techniques for Visualising Fire Behaviour. In: *T. GIS* 11 (2007), Nr. 4, S. 621–635. http://dx.doi.org/http://dx.doi.org/10.1111/j.1467-9671.2007.01063.x. – DOI http://dx.doi.org/10.1111/j.1467–9671.2007.01063.x

[7] PANG, Alex: Visualizing Uncertainty in Natural Hazards. In: *Risk, Governance and Society* 14 (2008), S. 261–294

[8] ANDRIENKO, Gennady L. ; ANDRIENKO, Natalia V. ; DYKES, Jason ; FABRIKANT, Sara I. ; WACHOWICZ, Monica: Geovisualization of dynamics, movement and change: key issues and developing approaches in visualization research. In: *Information Visualization* 7 (2008), Nr. 3-4, S. 173–180

[9] BLOK, Connie: Monitoring change: characteristics of dynamic geo-spatial phenomena for visual exploration. (2000), S. 16–30

[10] PETERS, Ellen ; DIECKMANN, Nathan ; DIXON, Anna ; HIBBARD, Judith H. ; MERTZ, CK: Less is more in presenting quality information to consumers. In: *Medical Care Research and Review* 64 (2007), Nr. 2, S. 169–190

[11] KUNZ, Melanie ; GRÊT-REGAMEY, Adrienne ; HURNI, Lorenz: VISUALIZING NATURAL HAZARD DATA AND UNCERTAINTIES–CUSTOMIZATION THROUGH A WEB-BASED CARTOGRAPHIC INFORMATION SYSTEM. In: *Proceedings of the special joint symposium of ISPRS Techn. Commission IV & AutoCarto 2010* (2010)

[12] JIANG, Carl Y.: Modeling Bushfire Spread Based on Digital Elevation Model and Satellite Imagery: Estimate Burning Velocity and Area. In: *American Journal of Geographic Information System* 1 (2012), Nr. 3, S. 39–48

[13] WIKIPEDIA: *Haversine formula — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Haversine_formula&oldid=553768263`. Version: 2013. – [Online; accessed 9-June-2013]

[14] CATMULL, Edwin ; ROM, Raphael: A class of local interpolating splines. In: *Computer aided geometric design* 74 (1974), S. 317–326

[15] GRAHAM, Ronald L.: An efficient algorith for determining the convex hull of a finite planar set. In: *Information processing letters* 1 (1972), Nr. 4, S. 132–133

[16] GOLDMAN, Ronald: Graphics gems. (1990), 304–. `http://dl.acm.org/citation.cfm?id=90767.90838`. ISBN 0–12–286169–5

[17] DIJKSTRA, Edsger W.: A note on two problems in connexion with graphs. In: *Numerische mathematik* 1 (1959), Nr. 1, S. 269–271

[18] TOLHURST, Kevin ; SHIELDS, Brett ; CHONG, Derek: Phoenix: development and application of a bushfire risk management tool. In: *Australian Journal of Emergency Management, The* 23 (2008), Nr. 4, S. 47