Universität der Bundeswehr München

Fakultät für Informatik

Institut für Theoretische Informatik, Mathematik

und Operations Research

A Decision Support Tool for Wide Area Search Missions based on Predictive analytics

-An intuitive visualization of environmental information collected by drones-

Robin Westphal

Providing Professors:

Prof Dr. Stefan Pickl & Prof. Dr. Alexander Bordetsky

Master Thesis

August 2017

Content

List of Figures

List of Tables

Acknowledgments

I would like to thank everybody who supported me in writing this master thesis at the Center for Network Innovation and Experimentation (CENETIX) at the Naval Postgraduate School (NPS) in Monterey.

My very special thanks are addressed to Prof. Dr. Alexander Bordetsky, head of CENETIX at NPS, as well as Prof. Dr. Stefan Pickl, who allowed me as a student of the University of the Federal Armed Forces in Munich, to write my thesis at a high-level institute in the United States, and for the possibility to work in an interesting area of operations research.

I also thank Steven Mullins, who always helped me with all his abilities. He supported me solidly in academic and technical questions of my research, administrative procedures and showed me many funny and interesting things to do in California.

Eugene Bourakov helped me also a lot with his technical expertise and Manon Raap supported me with her experience in this research field.

I would also like to thank Dr. Martin Zsifkovits and Dr. Michael Preuss, who offered several good ideas for the general structure of this thesis.

In the end, I would like to thank my wife, Désirée, who supported me during our stay in the United States of America.

1 Introduction

The research presented in this thesis was initiated by the Naval Postgraduate School, USA, and the Universität der Bundeswehr, Germany. One tutor, Prof. Bordetsky, is working at the Department of Information Sciences. In this department is a research effort for wide area search missions with unmanned vehicles and systems. This department is in close contact with Prof. Pickl, who is head of the Institute for Theoretical Computer Science, Mathematics and Operations Research at the faculty of computer science at the Universität der Bundeswehr in Munich, Germany.

Both institutes work together closely to improve existing algorithms and to develop new procedures and tools to solve diverse optimization problems.

This thesis is based on existing work, which deals with path-optimization, network communication and wide area search and connects these fields to create an easy to use User Interface. The objectives of this work are specified at the beginning. Subsequently, definitions and concepts are introduced which are important for this work. In addition, the tool is explained. The main section will explain the model of this tool and the mathematical background in detail. After this a summary and evaluation of the tool are presented where challenges and problems are discussed in detail.

Finally, we discuss, how this tool can be improved, in which areas it can be used and where it will be used in the near future.

1.1 Motivation

Wide area search missions are very important. They are necessary to save the lives of people or to prevent them from harm. The current situation in the Mediterranean and Aegean seas due to the refugee crisis are examples, where search and rescue missions could have saved the lives of many people. In 2015 at least 2500 people died after a refugee ship sank [1] and over 5000 people drowned in 2016 [2].

But wide area search missions are not limited to maritime rescue missions. They are also employed to rescue as many people as possible in several other disasters such as the tsunami in 2004 which cost the lives of more than 230,000 people [3], or the earthquake in Haiti in 2010 where also 230,000 people died [4].

Wide area search missions can also be used to observe and prevent areas from harm. When an acute threat situation is known it is possible to observe the threatened area to find and locate an attacker and thwart his plans.

Those tasks are often supported by systems based on theoretical search approaches, e.g., the United States Coast Guard uses a search and rescue planning system [5]. But often the searchers need to plan the path by hand, which is a tremendously complex process.

It is confirmed to be an $\mathcal{NP}$-complex problem by [6] for a single platform searching for a single stationary target. In many search missions, several more platforms are searching and often the search target can move around or more than one target exists. Generally, this leads to more complex missions. Moreover, many more restrictions must be considered. Like kinematical constraints of the searcher depending on his vehicle and the search area and constraints on resources. Also, a search mission is usually time critical.

To be able to lead a complex search mission it is necessary for the coordinator to have a good situational picture. In this picture, all available data should be taken into account and it is important that the data is represented in a way that is fast and easy to understand [7].

1.2 Goal

Various objectives are pursued with this thesis. The main goal is to develop a good management interface, so the user can easily observe an area of interest. Although it is irrelevant what type of search is conducted, the idea of this work is to generate a tool for search missions with unmanned vehicles.

In addition to the pure visualization, the observer should also be able to coordinate and control the search mission based on current data. To do this he should be able easily adjust the restriction of the search mission. For this, an intuitive interface is offered in which he can specify restrictions and expected start conditions.

In the context of predictive analytics, it should be possible to create simple behavioral patterns. This behavior can lead to new restrictions and can have a massive influence on the search history. For example, the user should be able to specify how an item of interest is likely to move over time. With this feature, it is possible that the visualization not only shows when and where an object has been spotted, but also how likely it is that this object is located in another location after a certain time duration.

A specific application might be the search of shipwrecked people. If the user knows the location and the time of the accident and knows something about the current of the sea in the vicinity of the accident, this tool can be used to estimate where the shipwrecked people probably are after a certain time. Based on this the search teams or drones can be automatically sent to the appropriate areas.

1.3 Definition of terms

Several specific terms are used in this thesis. The reader will often encounter terms, that are related to specific geographic areas. An “area” is a certain territory with set boundaries. A “subarea” is a part of a greater area.

The largest area is an area of interest. Within this area the user searches for something, the so-called item of interest. Every subarea, which is part of a specific area of interest, is always a proper subset to this area of interest. The maximum expansion of a subarea can only be as large as its area of interest. There may be several types of subareas within an area of interest. One subtype is a smaller area, where the user can define other initialization values. The most important subareas relate to the specific behavior of an item of interest within an area of interest. Another important subarea type is a smaller field for the visualization purposes.

It is possible to designate several areas of interest at the same time. These areas can overlap, but each subarea can have only one related area of interest.

An item of interest is something that the user tries to locate and observe within a specific area. The item could be one of many possible things. It could be a radioactive source like a dirty bomb, which is hidden somewhere; it could be a missing person who is lost at sea and gets moved around by ocean currents; or it could be a terrorist, who carries explosive devices and moves through a built-up area trying to get to a specific location.

Another term is “sensor”. Sensor describes any device, that is capable to detect an item of interest. The actual type of sensor depends on the item of interest. It could be something that finds heat signatures to locate the missing person at sea. It could a technical gadget that is able to detect radiation or it can be a person who is looking for someone.

The sensors may be carried by manned or unmanned vehicles which can operate on land, in the air or at sea.

2 Literature Review

This chapter will outline the theoretical foundations. To do this each fundamental work which is used in this thesis will be discussed and the current status of the research in this field will be explained. Each field will be only discussed in a level of detail that is necessary to understand why this research field is applicable for this thesis. The relevant research fields are decision support, predictive analytics, management information system, searching under constraints and moving target search. At the end, a closer view is given to the software applications that were used.

2.1 Decision Support System

Decision Support Systems (DSS) describes computer-based systems that help decision makers. To do so they need to filter, identify, process and visualize the necessary data for humans. Decision Support Systems have many functionalities depending on the field in which they are used. Some systems may only do simple calculations to find some average values. Other systems comprise complex models which try to evaluate lots of data and find possible actions.

Decision making is still a task for humans. The system only serves to help the decision maker by displaying complex data or finding special features.

It is important for a Decision Support System that it can be easily adjusted towards specific problems and issues. It should also work automatically in order to decrease the effort to use it as much as possible [8, 9].

There are different taxonomies for Decision Support Systems. Haettenschwiler [10] distinguishes between active, passive and cooperativ DSS. An active system can suggest explicitly while a passive system can only provide useful information. A cooperative system allows an iterative process between the user and the tool to improve the suggestions of the tool. Daniel Power [11] differentiates between the input type of the DSS. He speaks of communication-driven DSS, where the user or several users can directly communicate with the DSS; of data-oriented systems, where the tool manipulates data; and of document-driven DSS, where the system manages, retrieves and manipulates unstructured information. He also describes two categories where the system is driven by facts, rules or procedures; the knowledge-driven DSS and where the DSS is based on a model, the model-driven DSS.

Another interesting approach to classify different DSS made by Power is to differentiate between enterprise-wide DSS, where the system is linked to a large data warehouse and can be used by several persons within a company, and a desktop DSS, where only a small system is running. Usually only for an individual user.

Especially in the military many Decision Support Systems are used. In the military, it is often necessary to make decisions under time pressure, based on insufficient data. Compared to decisions that are made in a civilian company, these decisions endanger directly the life of soldiers. Due to this peculiarity, this research field is often applied and modern computer based approaches are used to improve the decision making in the military [12, 13].

Many different approaches are used to develop a complex Decision Support System. Usually it involves developing scenarios and creating complex models. This leads to many difficult methodological problems. Many factors in such a system are non-quantifiable, cannot be described properly or delineated completely. Furthermore, a complex problem is usually affected by many inherent uncertainties and many factors are not known. However, there are some methods and techniques which try to structure the development of such a system [14, 15, 16].

2.2 Predictive Analytics

Predictive analytics are a number of techniques and tools that analyze current and historical data to forecast about specific events in the future. Usually a wide number of statistical techniques and ideas of predictive modelling, machine learning and data mining are used to make this forecast [17, 18].

Predictive analytics can help to optimize existing processes. They can help to understand specific behavior and are able to identify opportunities and to anticipate problems. Wayne Eckerson defines predictive analytics as *a set of business intelligence (BI) technologies that uncover relationships and patterns within large volumes of data that can be used to predict* *behaviors and events. Unlike other BI technologies, predictive analytics is forward-looking, using past events to anticipate the future. (See figure 1)* [19]

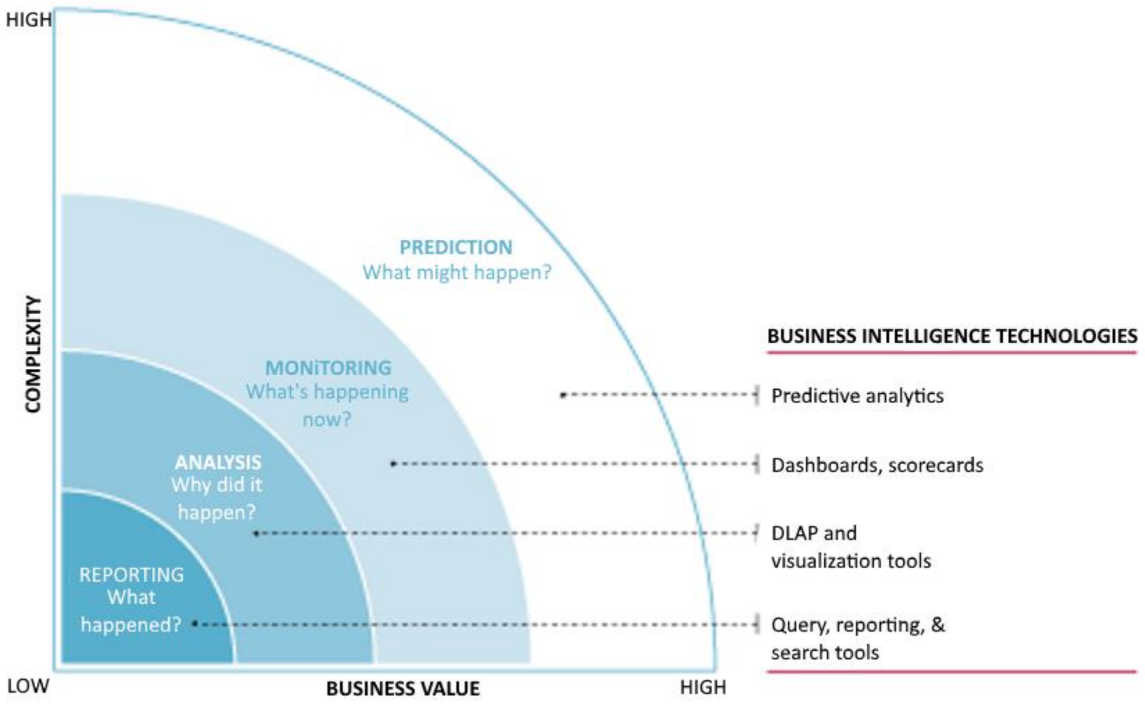


Figure 1 Predictive analytics in business intelligence [19].

Eric Siegel describes predictive analytics as an area of statistics that deals with extracting information from data and uses this information to predict trends and behavior patterns. Predictive analytics are usually used forecast an event of interest in the future but can be also used to understand an unknown incident in the past or present [20].

Predictive analytics are used in many applications. One of the biggest fields is in the finance sector [21]. The technical analysis is a methodology for forecasting the direction of prices on charts based on current and past data of the market [22]. This analysis is often used at the stock exchange. Even when it is not proven that it is impossible to forecast the prices at the stock exchange and the technical analysis is a direct contradiction to the Random Walk Theory from Karl Pearson [23] or the Finance Market Theory from Patrick Wegmann [24]. One of the biggest argument to use predictive analytics in the finance sector is that it could be a self-fulfilling prophecy when enough people use these techniques.

Another research field for predictive analytics is in marketing, to find and forecast typical behavior of customers and to be able to identify trends and demands [25]. But predictive analytics are also used in healthcare [26], mobility [18], telecommunications [27] and many other fields.

Three types of predictive analytics can usually be distinguished. All these types are often called predictive analytics, but the statistical techniques and the necessary data are very different depending on the purpose of the model. The first type are predictive models. The object of this model is to find the likelihood that a specific event will happen with a specific impact, based on known similar events. An example of this is to analyze the blood splatter in a simulated crime scene to find how the blood splatter in the real crime scene was created [28].

The second type are descriptive models. Unlike predictive models which try to predict specific behaviors, descriptive models identify different relationships between objects of the model. For example, they try to find relationships between customers and products and to classify the customers based on the discovered relations [29].

The last type are decision models. These models try to describe all necessary relationships between all elements of the model in order to predict the results of certain decisions which are made within the model. They can be used in optimization. Usually they are used to develop decision logic or rules to simulate the outcome of events within a specific research field, like the shopping behavior of customers based on the placement of advertisements [30].

Many techniques can be used for predictive analytics. They can be distinguished between regression techniques and machine learning techniques. Regression techniques are often used in statistical modeling and they are tools to modeling and analyzing several dependent and independent variables. They could be used to describe quantitative relationships and to forecast the value of dependent variables [31, 32]. Examples of these techniques are the linear regression model, discrete choice model and logistic regression [33].

Machine learning describes a subfield of computer science where computers try to learn autonomously patterns based on specific input data [34, 35]. The most common applications are based on neural networks [36].

The database is essential for predictive analytics. It is necessary to have enough data and the data needs to have sufficient quality. When there is not enough data it can be impossible to find or forecast the wanted patterns. When the quality of the data is not good enough it can happen that the assumptions are wrong [37]. The database of predictive analytics is in close relationship to tools used in data mining and big data tools [38].

Many people are skeptical about the ability to predict the future with computers based on algorithms. Gary King said that the viewed elements are influenced by many variables which are likely to change in many ways, and that it is necessary to know all these variables and measure them accurately. *"All of those variables are unpredictable. How they will impact a person is even less predictable. If put in the exact same situation tomorrow, they may make a completely different decision. This means that a statistical prediction is only valid in sterile laboratory conditions, which suddenly isn't as useful as it seemed before."* [39]

In a study from 1990 to 2006, over 1072 papers published in the top-rated journals *Information System Research* and *MIS Quarterly* were searched. Out of these papers only 52 had predictive claims and only seven of them carried out proper predictive modeling or testing [40].

Based on this a user of a tool which uses predictive analytics should be aware that the tool can only assume something and it is not guaranteed that this assumption is correct. It depends on the data provided and the accuracy of the model. However, even when it has worked perfectly before it is not guaranteed that it is still working well because variables in the real world could have changed.

2.3 Search Problem

In this thesis, a search problem describes a problem where it is necessary to find an object or place within a geographic area. This can happen in an n-dimensional room or in a graph-structure. For the user, it is more intuitive to understand the search algorithm when he can see it acting in a two or three-dimensional room, but it is easier to create a search algorithm where the data is based on a graph structure. A search algorithm is any algorithm which is capable to solve the search problem.

2.3.1 Vehicle routing problem

One of the most general search problems is the vehicle routing problem (VRP), where the start, destination and the environment (graph, search room, etc.) are well known. The VRP is a generalization on the travelling salesman problem (TSP). Usually the object of a VRP is to find an existing route and to minimize the total route cost. Determining the optimal solution to VRP is $\mathcal{NP}$-hard [41].

There are many search algorithms which seek to find fast acceptable solutions for a specific VRP. Each algorithm has specific focal points and there are algorithms which are able to solve the VRP very well but which can hardly be used on other problems. A greedy algorithm would be able to find a solution as fast as possible, but it is not guaranteed that the solution is good [42]. On the other hand, a Dijkstra's algorithm will always find the best solution but in a worst-case scenario it searches the whole room [43]. Some other algorithms try to combine the approaches. For computer games, the A* search algorithm is widespread [44]. For some specific search problem like the TSP, some search algorithms which try to be optimal for this specific problem are developed [45]. It is also possible to use modern heuristics like evolutionary algorithms [46] or swarm optimization approaches [47].

2.3.2 Routing under constrains

A search problem becomes more complex when constrains are considered. Depending on the actual search problem many constraints need to be taken into account. Usually there some resource constraints, the optimization criterion can change, or more than one criterion needs to be optimized and the criterions preclude each other. As an example, for the TSP it is possible that more than one salesman is moving around, the salesmen have only a specific time and can get only to a certain number of cities, the costs to move between cities depends on the track, the vehicle, the goods a salesman is carrying.

When constraints for a search problem exist, it is necessary to take them into account, otherwise it is likely to create an invalid solution.

It is also possible that the vehicle routing problem is dynamic. For example, it is possible that the graph changes over time, or the graph could be unknown at the beginning of the search or the position of the target is unknown. All these things can also be deterministic or stochastic [48]. One example for a search problem with an unknown position of the search target is given in [49].

2.3.3 Moving target

The next increase in complexity would be with a moving target. The problem of a moving target search can also be divided into several subgroups, depending on the number of platforms which are searching for the target and under kinematical and resource constraints.

A first approach to the moving target problem is given in [50]. Ishida and Korf worked out that it is necessary during a search like this that the searcher should be able to move faster than the target. Otherwise the target could evade the searcher indefinitely. They used a learning-real-time-A* (LRTA*) [51] to solve this problem. Other approaches are given in [52] and [53]. Here the search area is modeled by a finite number of cells C and the duration of the search is described as a sequence $K = (1, \dots, k)$ of k timesteps. The target can only occupy one cell $c \in C$ at a time step $k \in K$. During a search mission, a probability map p is maintained, where the likelihood $p_{k,c}$ that the target is at a specific cell c at the time k evolves over time. The initial position of a target is unknown and the target track is modelled by a stochastic process $(c_1, \dots, c_k)$, which is assumed to be Markovian [54]. A sensor z would be able to detect a target, when its position at time k is the same position as the target at time k.

An extension to the first moving target problem would be kinematical constraints. Depending on these constraints the target and the sensor can move only to another position c in one timestep depending on the distance between the current position and the next position. Without these constraints, it would not be possible to make a systematic search, because the target and the sensor could randomly jump to any cell $c \in C$. When those constraints are added it is necessary that the speed of the sensor or the number of cells where the sensor can go are greater than those of the target. Otherwise the same problem as described in [51] can occur.

The next increase in complexity would be resource constraints. In practice, these constraints could be interpreted as the fuel of the sensor or his capability to communicate. A resource limit could be described as $T \in \mathbb{R}^+$ and the required resource at cell c at time k is $r_{c,k} \in \mathbb{R}^+$ and the total consumption is limited to $T \geq \sum_{y=1}^{y \leq K} \sum_{x=1}^{x \leq C} r_{x,y}$.

The sensor model could also be changed, so it gets a glimpse probability $g_{c,k}$. With such a probability, it is possible to model the sensor in a way that it is able to oversee the target or to make a false assumption and detect the target even when it is not there. These constraints and assumptions are rational when a model like this should be used in a practice field. In a real-life search and rescue mission of shipwrecked people it is possible that some people might be overlooked or that the searcher thinks he saw someone when he actually saw an object and not a person.

Another practical approach would be a model with more than one search platform. In a real search and rescue scenario usually more than one searcher is trying to find the target. Where each sensor z is part of a set of sensors Z, these sensors could be homogeneous and use the same glimpse probability and resource constraints. It would also be possible to have heterogenous sensors where each has its own glimpse probability and resource constraints function.

2.4 Software Applications

The tool which is created in this thesis uses several software applications. In this section, a short glimpse about the used tools is given.

2.4.1 JavaScript

JavaScript is a scripting language which was invented in 1995. It is used to expand the possibilities of HTML and CSS. It is client sided and can be used to evaluate user input and to reload or change content without reloading the whole webpage or loading another webpage.

All modern browsers support it without the need for plug-ins. They have a built-in JavaScript engine. All these engines are based on ECMAScript specification. Some of them do not support the full specs of ECMA and many support additional features. ECMAScript is a trademark scripting language specification to standardize JavaScript [55].

JavaScript is a high-level programming language. This means it has a strong abstraction from the details of a computer. It is a dynamic programming language so it can execute behaviors during runtime like extending the program by objects. It is weakly typed, so the allocation of variables has no type based restrictions. And it is an object based language, so methods and states are encapsulated inside objects. It is not an object-oriented language because it does not support inheritance or subtyping [56].

2.4.2 NodeJS

NodeJS was developed originally in 2009 and is an open-source, JavaScript run time environment. In the initial release, it supported only Linux but it has evolved to a cross-platform. Historically JavaScript was only a client-side scripting language, where the program code is embedded in a HTML page. NodeJS enables developers to use JavaScript on the server side [57, 58].

"Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient" [59]

The reason, why NodeJS is used for this tool is that NodeJS is non-blocking [60]. This means that the server can handle several communications at the same time. And this is necessary for the tool, because the server needs to access the database very often.

2.4.3 Active Server Pages

Active Server Pages is a technology that is used on the server side and was developed by Microsoft. When a browser makes a request on an ASP file, the server can execute the script from this file using an ASP engine. After that execution, the server can send back a normal HTML file to the browser. [61, 62, 63]

The reason why ASP is used is that access to the database will be faster and easier via an ASP file. Otherwise the NodeJS server needs to deal with additional requests which could be easily outsourced to an ASP file.

2.4.4 MySQL

MySQL is one of the most used databases [64]. It is a relational database which describes a database model where the information is stored in tables. Relational databases were developed in 1970. The management system for these databases is called a relational database management system and usually SQL (Structured Query Language) is used to operate a database like this [65, 66].

A relational database is a collection of tables where each table has a specific number of entries. Each row is a separate entry and each entry has a defined number of attributes. The attributes can be seen as the columns of the table. Each entry has also a primary key so it can be clearly identified. It is necessary that every entry in the database is unique. It is also possible to have several primary keys for one entry. These multiple keys are known as a composite primary key [67].

2.5 Related Work

One of the first computerized decision support tools which was used for search and rescue missions was the Computer Assisted Search Planning (CASP). This tool was put in operation in 1974 by the U.S. Coast Guard. It used a Bayesian approach to produce probability distributions for the location of a search object [68, 69]. The U.S. Department of Homeland Security also developed also a user manual for the U.S. National Search and Rescue Supplement (NSS) [70] which is based on the International Aeronautical and Maritime Search and Rescue Manual (IAMSAR) which was developed by two specialized agencies of the United Nations, the International Maritime Organization (IMO) and the International Civil Aviation Organization (ICAO) [71, 72, 73].

In 2003 the U.S. Coast Guard started to develop a Search and Rescue Optimal Planning System (SAROPS) as a new decision support tool for search and rescue missions. This tool has operated since 2007 and has three main parts. A graphical user interface which uses Environmental System Research Institute (ESRI), Geographic Information System (ArcGIS) and U.S. Coast Guard specific applications such as search and rescue (SAR) tool extensions. Vector and raster charts are available to display plans, patterns, probability maps and environmental data.

An Environmental Data Server (EDS) collects and stores all available environmental data to use within SAROPS. For example, it stores data for certain geographical areas about the temperature of the sea and the air, wave height, tides and model output from operational forecast models like the hybrid coordinate ocean model (HYCOM).

The last core component is the SAROPS Simulator (SIM). The simulator uses available information about the search object and the environment to produce probability distributions for the object's location. It is also based on Monte-Carlo simulation [68].

Another approach of wide area search decision tools is given in [74]. This approach is used to assist commanders, operators and planners in special operations missions. It tries to find locations where UAVs could effectively be deployed and employed within an area of interest where a special operations team is assigned to search and detect targets. The model prescribes optimal deployment locations for the ground units and optimal time-phased search areas for the UAVs.

3 Methodology

In this chapter, the tool will be discussed in detail. At the beginning a short overview about the tool is given, showing the functionality of the tool. After that the model of this tool is explained in detail. Especially the mathematical background of this model is discussed. In the end, the implementation of the tool is shown.

3.1 Purpose of the tool

As mentioned, the goal of this work is to create a decision support tool for a wide area search mission. The tool should work independently of the actual method of search and should be able to improve the searching behavior, by providing the operator a good overview of the search area.

This tool should also be able to forecast possible status of the position of an item of interest within the observation area. To do this it uses specific data which are collected by drones and uses this data and given rules to calculate the possible states. The data will be visualized in an intuitive understandable management information system, so the user can easily use and understand this data.

It should be possible for the user to define the settings for a specific search mission. He should be able to specify the search area, to initialize this area with known or assumed data and to create simple logic behaviors which are used to make forecasts.

The user can employ this tool with a web client in any modern browser. The data should be stored on a server in order to enable have multiple clients monitor the area.

An example where this tool would be useful is proposed in [52, 53]

3.2 The Decision Support Tool

With this decision support tool, the user can monitor a certain area of interest. To do so he can define it in 2-D space specify simple logic behavior for it. To do this the tool uses various web pages, a server, an active server page and a database.

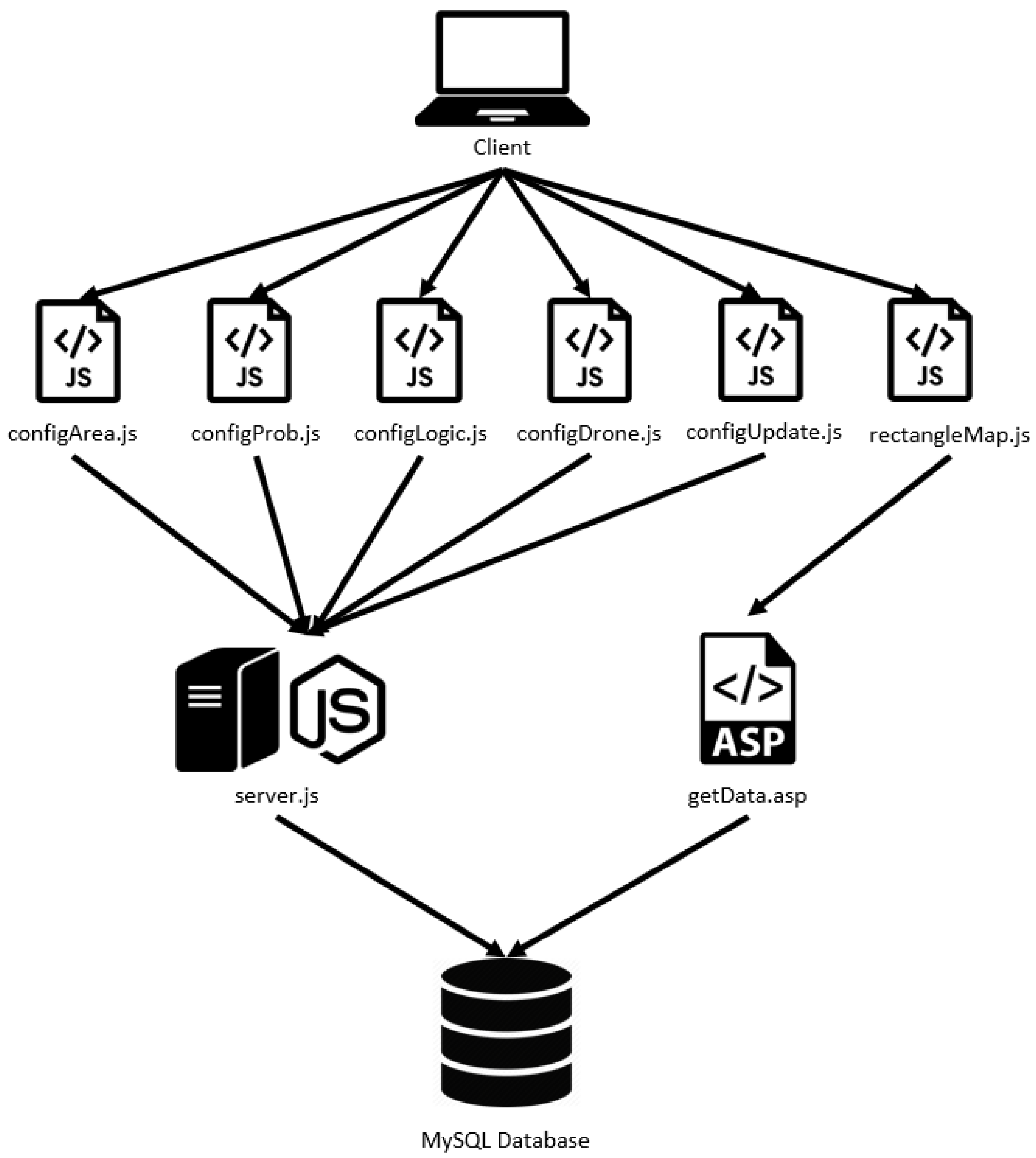


Figure 2 Structure Diagram

The structure of this tool is given in figure 2. The client can access several config files and a file for the observation mode. In the config files, the user can set and modify several things. The config files communicate directly with the server. All the calculations based on the desired logical behavior are made on the server and, which stores and manipulates all relevant data on a MySQL Database.

To observe the area the user can access the showMap.js file. This file will display all relevant data. To do this it communicates over an active server page with the database to retrieve all needed data. In figure 3 the visualization of this tool is shown.

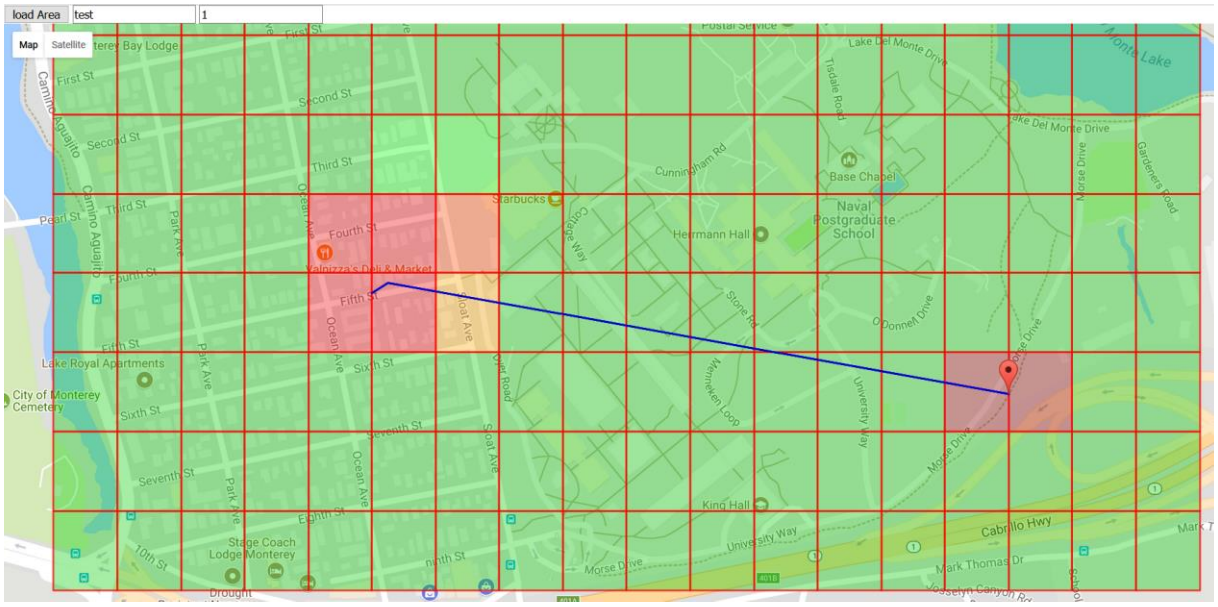


Figure 3 Show map

First a small overview of every file is provided. Then, the model is explained in more detail. In the next chapter, the implementation will be discussed. A user guide is provided in the annex.

When the user wants to observe an area of interest, he can use the showMap.js file. This file will display a Google Map. On this map, the observation area is drawn. The different colors show the probability that an item of interest is in a particular subarea. When several areas are observed at the same time, the user can select a certain area. Only this area will be displayed in the window. When he wants to observe another area at the same time he constructs a different window.

When the user zooms in and out on Google Maps the view will adjust automatically. He can also adjust the displayed size of the rectangles. When a greater size is used the computer requires less calculations and less communication with the database and should run faster. On the other hand, the graphic representation will be less precise.

All active sensors for this area will also be displayed. The path of a sensor is shown as a blue line and the last known position of the sensor is marked with a dot. When the user hovers over this dot with a mouse, the name of the sensor is displayed. As shown in figure 4.

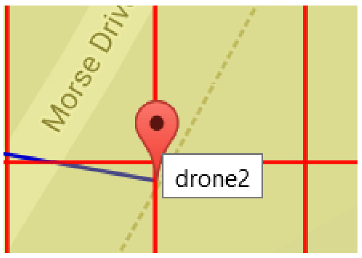


Figure 4 Sensor name

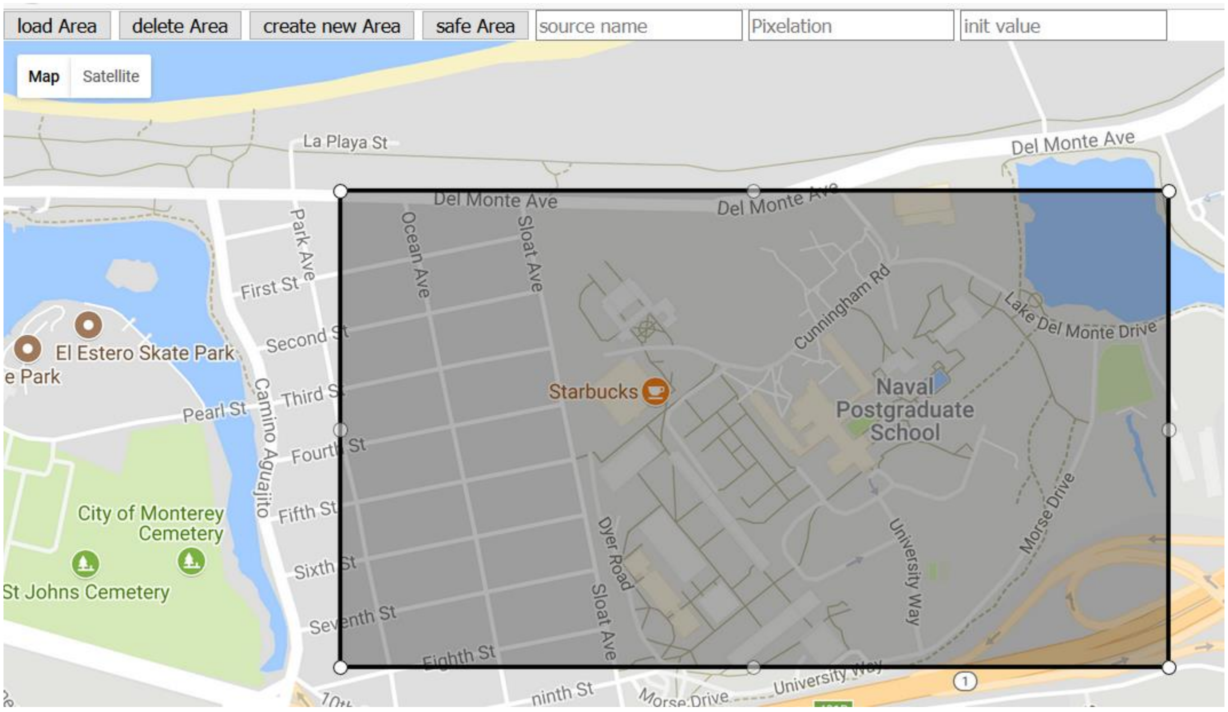


Figure 5 Set map

In the configArea.js file the user can create an observation area. He can set the pixelation and define a name for this area. The pixelation defines the accuracy of the observation area and is based on the latitude and longitude values. This will be explained in more detail, after the coordinate system is discussed. Upon creation, the area consists of several cells (figure 5).

The area name is used to find it among the other config files and to make further adjustments to it. The name is also used to display the area in the showMap.js file and to assign sensors to it.

The user can also set an initialization value for the whole area. This value represents the possibility that an item of interest is inside this area. The possibility can also be adjusted in the configProb.js file and is affected by the logic behavior and the data of the drones. Each cell requires this initialization value.

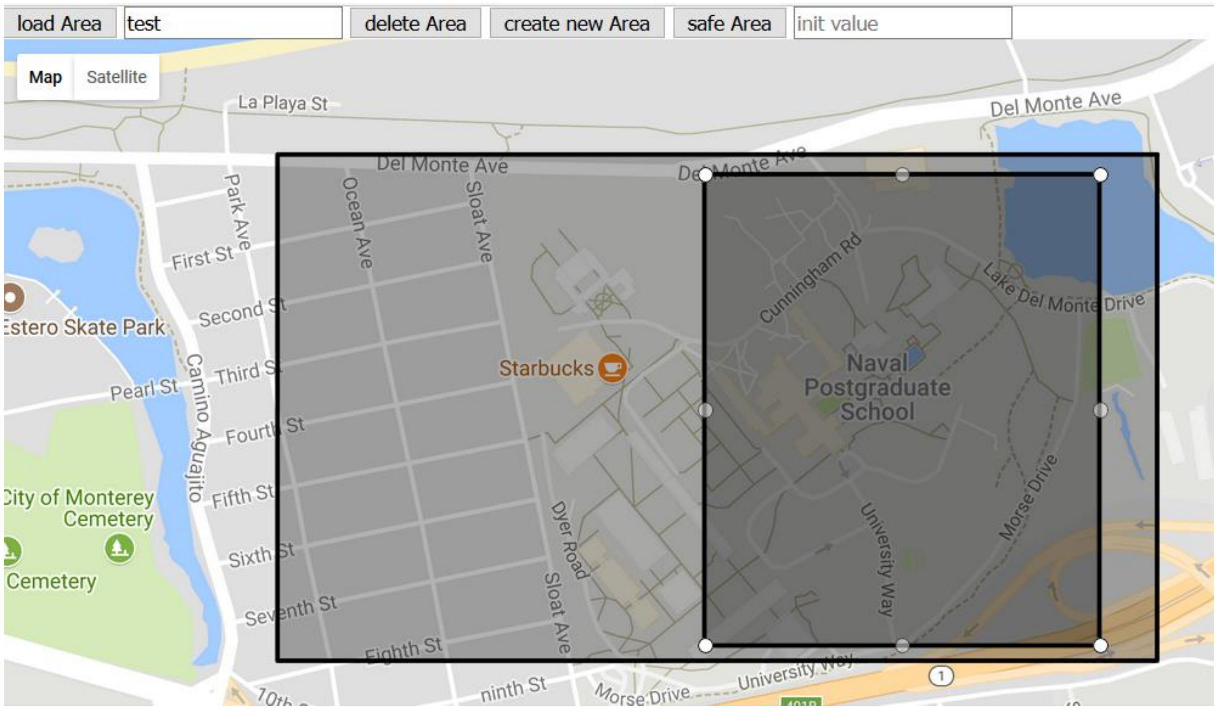


Figure 6 Init map

In the next config file, the user can set the initialization values more precisely. He can create a subarea within the observation area and can set the probability that something is in this subarea.

He can only adjust cells within the observation area. When he creates a subarea outside of the main area of interest the values will be ignored. The old values will also be overwritten. After saving a subarea with a specific value the old values will be lost. There can be only one subarea at a time, but it can be moved and its values changed as often as required (figure 6).

When the user wants to assign sensors to a certain area he can do this in the configDrone.js file. How it will look like is shown in figure 7.

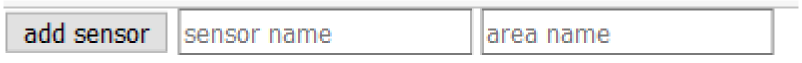


Figure 7 Add sensor

To do so he needs to name the sensor and its area of interest. A table with the name of the sensor is created in the database. The sensor needs to update this database by itself directly.

The user has also the possibility to delete areas of interest and sensors.

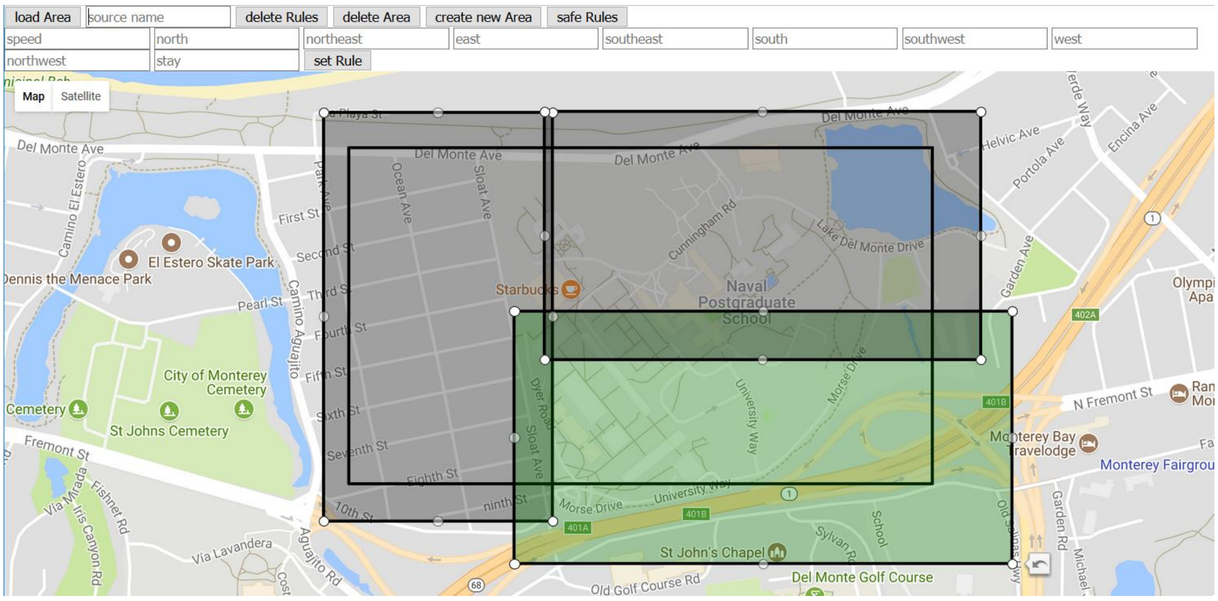


Figure 8 Set logic

The user can also create simple logical behavior for the defined area of interest in the configLogic.js file. Here he can create several subareas (figure 8). For each subarea, he can define the speed at which the item of interest may travel and the likelihood, that an object stays at its current position or moves in a certain direction (north, northeast, east, southeast, south, southwest, west and northwest).

This can be done for every area of interest. When the subareas of two rules overlap, only one rule will be used.

In the last file, the user can activate the rules for an area. By default, the rules are not activated. After the rules are activated they will be applied for this area (figure 9).

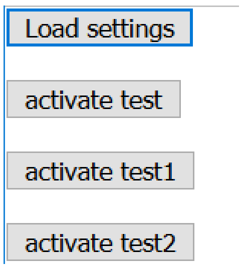


Figure 9 Set Update

The position and areas in this tool are based on latitude and longitude values. The most common way to define the terrestrial position is with two coordinates, latitude and longitude. This is the reason why latitude and longitude values are used to define certain areas and positions in this application. There are several different 'latitude and longitude' systems. The definition of several necessary values is a little bit different for each of these coordinate systems, like the prime meridians and equators. In Figure 10 are three points with the same latitude and longitude, but they use different coordinate systems (OSGB36, WGS84 and ED50) [75]. Due to this it is necessary to specify which coordinate system is used in a tool. So, a user can convert between different systems, if he uses data from other sources.

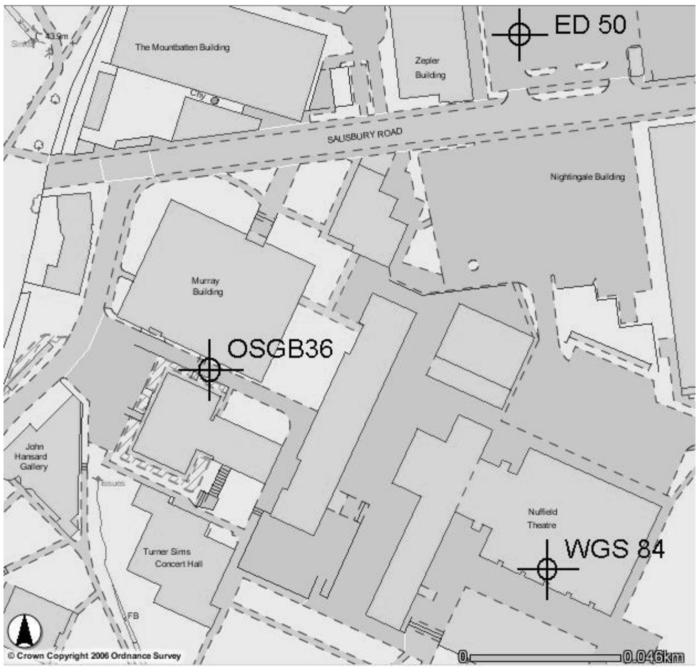


Figure 10 Coordinate systems

This tools default is the World Geodetic System 1984 (WGS84). This is an Earth-centered, Earth-fixed terrestrial reference system and geodetic datum. It is the standard U.S Department of Defense definition of a global reference system for geospatial information and is the reference system for GPS [76].

The user should also know that a latitude and longitude based approach to localize positions and determine distances can introduce several errors. An approach that is only based on these values can barely take different altitude levels into account. Calculations with a system like this use spherical geometry and assume the sphere as an optical sphere. The earth is not a well-shaped sphere due to different reasons. The spinning and rotation of the earth deforms the shape and the surface of the earth is also not uniform. Due to the size of the earth this effect can be neglected. The earth is flattened by 0.3% at the poles, and the size difference of the radius is -14km at the poles and +7km at the equator [77].

The actual position of a point based on latitude and longitude can also change over time. The earth is continuous deforming. The ground can move up and down over one meter per year just due to the gravitational influence of the sun and the moon, the same effect which causes the tides. The continental drift will change the relative position between two points over the years [78]. And there are several other phenomena, which change the surface of the earth.

But all of these effects happen very slow and will have usually a less impact on the distance calculation for this tool. But the user should be aware that small calculations errors can occur when the area of interest is close to one of the poles or some of the data is very old.

When the user sets a pixelation value of 0.001, the latitude of the north border of the smallest rectangles of an area of interest for this area will be 0.001 points greater than the latitude of the south border. This behaves in the same way for the east and west border. Each cell will have a north-south and an east-west extension of 0.001 latitude and longitude degrees.

3.3 Mathematical Model

As explained, the user can create observation areas. Each area a is described as a number of cells C with a specific position. The numbers of cells are based on the size of the area and on the given pixilation value pix. Each area has a north, south, east and west border. The north and south borders are latitude values and the east and west borders are longitude values. The pixilation value is a rational number and describes the latitude and longitude distance between two cells.

$$north\ border\ of\ area\ a\ =\ nb_{(a)} \tag{3.1}$$

$$south\ border\ of\ area\ a\ =\ sb_{(a)} \tag{3.2}$$

$$east\ border\ of\ area\ a\ =\ eb_{(a)} \tag{3.3}$$

$$west\ border\ of\ area\ a\ =\ wb_{(a)} \tag{3.4}$$

$$pixelation\ of\ area\ a\ =\ pix_{(a)} \tag{3.5}$$

$$latitude\ of\ north\ border\ =\ lat_{(nb_{(a)})} \tag{3.6}$$

$$latitude\ of\ south\ border\ =\ lat_{(sb_{(a)})} \tag{3.7}$$

$$longitude\ of\ east\ border\ =\ lng_{(eb_{(a)})} \tag{3.8}$$

$$longitude\ of\ west\ border\ =\ lng_{(wb_{(a)})} \tag{3.9}$$

$$lng_{(eb_{(a)})} > lng_{(wb_{(a)})} \wedge lat_{(nb_{(a)})} > lat_{(sb_{(a)})} \tag{3.10}$$

$$|C|_{(a)} := \left\{ \begin{matrix} i * j : \; i, j \; \in \mathbb{N} : i = \dfrac{lng_{(eb_{(a)})} - lng_{(wb_{(a)})}}{pix_{(a)}} \\ : j = \dfrac{lat_{(nb_{(a)})} - lat_{(sb_{(a)})}}{pix_{(a)}} \end{matrix} \right\} \tag{3.11}$$

Each cell has its latitude and longitude and their position compared to other cells in the area based on a Euclidean grid. The latitude and longitude are used to display an area based on the cells inside of it and to make distance calculations between cells. To access cells faster, each cell has an x and a y position.

Each cell also has a value v, which is a number between zero and one. It describes the probability, that a specific source is inside this cell.

When the user creates an observation area the indices for the cell on its southwest corner equal zero. The x index of cell c' is greater than the x index of cell c, when cell c' is north of cell c. The y index is greater, when cell c' is east of cell c.

$$c \; \in C_{(a)} \tag{3.12}$$

$$cell\ on\ position\ x, y \; = c_{(x,y)} \tag{3.13}$$

$$x\ position\ of\ cell\ c \; = x_{(c)} \tag{3.14}$$

$$y\ position\ of\ cell\ c \; = y_{(c)} \tag{3.15}$$

$$latitude\ of\ cell\ c \; = lat_{(c)} \tag{3.16}$$

$$longitude\ of\ cell\ c \; = lng_{(c)} \tag{3.17}$$

$$value\ of\ cell\ c \; = v_{(c)} \tag{3.18}$$

$$lat_{(c)} > lat_{(c')} \Leftrightarrow \; x_{(c)} > x_{(c')} \tag{3.19}$$

$$lng_{(c)} > lng_{(c')} \Leftrightarrow \; y_{(c)} > y_{(c')} \tag{3.20}$$

The distance between two adjacent cells is based on user settings.

$$lat_{(c_{(x,y)})} = lat_{(c_{(x+1,y)})} + pix \tag{3.21}$$

$$lng_{(c_{(x,y)})} = lng_{(c_{(x,y+1)})} + pix \tag{3.22}$$

In the logic settings, the user can create a transition function tf to define the probability that a source is moving from one cell c to an adjacent cell c'. The chance that something is moving from cell c to c' cannot be greater than one and is at least zero. The sum for all probabilities how a source can move away from a cell, or stay at this cell equals one.

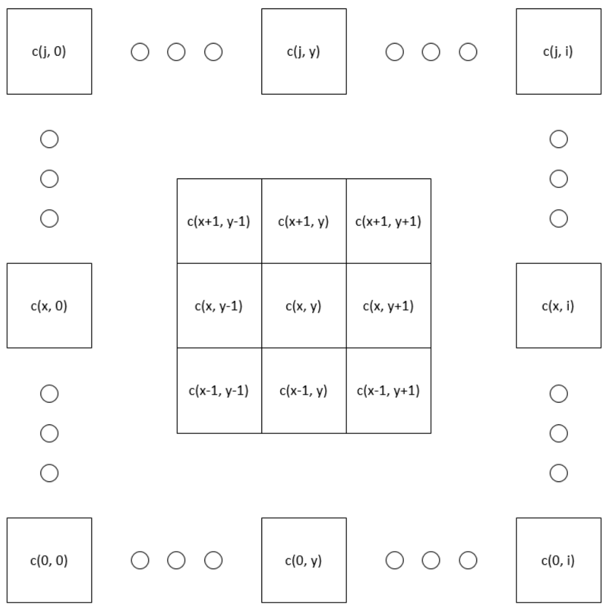
$$probability\ source\ is\ moving\ from\ c\ to\ c' \coloneqq tf(c, c') \tag{3.23}$$

$$tf(c, c') \in \mathbb{Q} \wedge \mathrm{tf}(\mathrm{c}, \mathrm{c}') \geq 0 \wedge \mathrm{tf}(\mathrm{c}, \mathrm{c}') \leq 1 \tag{3.24}$$

$$1 = \sum_{j=y-1}^{j=y+1} \sum_{i=x-1}^{i=x+1} tf(c_{(x,y)}, c_{(i,j)}) \tag{3.25}$$

This transition function is set for an area la. In this area, every cell uses the same transition function. The area la is always a part of the area a.

When the user creates an area for a specific transition function he can also set the moving speed of the source in this area. The speed will have two effects. The first one affects the update speed and will be explained later in the implementation section. The second affects the probabilities of the transition function. When the user describes the logic rule it is assumed that the source can move at the same speed in every direction. But the distance between two adjacent cells is not always the same. The cells are on a grid and each cell has eight neighbors.



$$x > 0 + 1$$
$$y > 0 + 1$$
$$j > x + 1$$
$$i > y + 1$$
$$lat_{(c_{(j,0)})} > lat_{(c_{(x,0)})} > lat_{(c_{(0,0)})}$$
$$lng_{(c_{(0,i)})} > lng_{(c_{(0,y)})} > lng_{(c_{(0,0)})}$$
$$lat_{(c_{(0,0)})} = lat_{(c_{(j,0)})} + pix * j$$
$$lng_{(c_{(0,0)})} = lng_{(c_{(0,i)})} + pix * i$$

And each cell also has a latitude and longitude position. Usually the distance in a Euclidean grid [79] system can be calculated with the Pythagorean theorem [80]. But the cells use only a Euclidean grid system to be accessed as rapidly as possible. The distances between the cells are based on their latitude and longitude values.

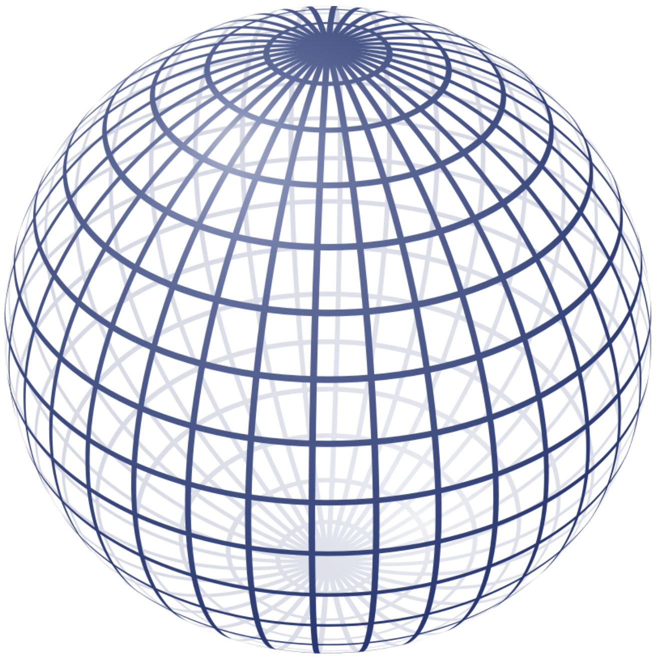


Figure 11 2-sphere wireframe as an orthogonal projection [81]

To make a distance calculation on a great sphere it is necessary to use spherical trigonometry [82]. The earth can be seen as a great sphere. The research field of geodesy is specialized on measurements and applied mathematics of the earth [83].

To consider these different distances between the cells the probability of the transition function will be changed slightly. In the first step, the distance between each adjacent cell is calculated. To calculate these distances the Haversine formula is used. The Haversine formula is based on the law of Haversines and is a special case of a general formula from spherical trigonometry. The formula is used for navigation and was first mentioned 200 years ago [84].

As proposed by Korn [85] the law of Haversines states:

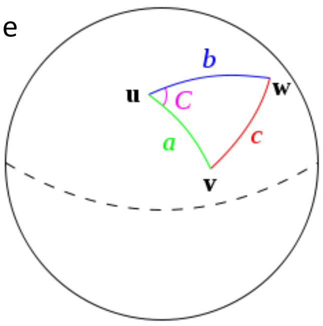


Figure 12 Law of Haversine [86]

$$hav(c) = hav(a-b) + \sin(a) * \sin(b) + hav(C) \qquad (3.26)$$

for the three points u, v, and w. The sides a (from u to v), b (from u to w) and c (from v to w) and C is the angle of the corner opposite from c.

Where hav is the haversine function:

$$hav(\theta) = sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2} \tag{3.27}$$

Based on the law the Haversine formula for the cells c and c' is:

$$hav\left(\frac{d}{r}\right) = hav\left(lat_{(c)} - lat_{(c')}\right) + \cos\left(lat_{(c)}\right) * \cos\left(lat_{(c')}\right) * hav(lng_{(c)} - lng_{(c')}) \tag{3.28}$$

Solving for d and applying the haversine function, the distance is:

$$d = 2r * \arcsin(sq) \tag{3.29}$$

$$sq = \sqrt{sin^2\left(\frac{lat_{(c)} - lat_{(c')}}{2}\right) + \cos(lat_{(c)})\cos(lat_{(c')}) * sin^2\left(\frac{lng_{(c)} - lng_{(c')}}{2}\right)} \tag{3.30}$$

Where d is the distance between cell c and c' and r is the radius of the earth.

When the distances are known, the mean distance to all adjacent cells of a specific cell is calculated and the probability will be adjusted by a factor which is based on the relations of these distances.

$$d_{mean} = \frac{\sum_{j=y-1}^{j=y+1}\sum_{i=x-i}^{i=x+1} d(c_{(x,y)}, c'_{(i,j)})}{8} \tag{3.31}$$

$$tf(c,c') = tf(c,c') * \frac{d(c,c')}{d_{mean}} \tag{3.32}$$

The formula to calculate the value of a cell after one timestep is based on the formula from Manon Raap [53].

$$pc_{k+1}(c) = \sum_{c' \in C} d(c',c)pc_k(c') \tag{3.33}$$

The target trajectory in this paper is modeled by a stochastic process and is assumed to be Markovian [54]. In this model, the movement is also modeled as a stochastic process and is based on a timestep. So, the value of each cell can also be seen as a Markovian chain in discrete time [87].

As an example, each cell has only two neighbors and a transition function which tells the probability that an object moves to one of these adjacent cells or stays at the current cell.

The transition function can be different for each cell. In this example are three different functions. The first rule describes the transition function for every cell with an index lower than x. The second function describes the behavior for the cell with index x and the last function is for every cell with an index greater than x.

	$tf(c_{(i)}, c_{(i-1)})$	$tf(c_{(i)}, c_{(i)})$	$tf(c_{(i)}, c_{(i+1)})$
i < x	0.2	0.2	0.6
i = x	0.1	0.8	0.1
i > x	0.6	0.2	0.2

Table 1 Transition function rules

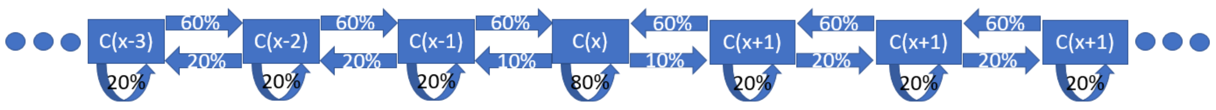


Figure 13 Transition function example

With the rules from figure 13, an object will move toward cell $c_{(x)}$. When the calculation for the value of cell $c_{(x)}$ is made, the sum of the probabilities that something moves or stays in this cell equals 2.

$$tf\left(c_{(x-1)}, c_{(x)}\right) = 0.6\,; tf\left(c_{(x)}, c_{(x)}\right) = 0.8;\ tf\left(c_{(x+1)}, c_{(x)}\right) = 0.6 \tag{3.34}$$

When a basic Markovian chain approach is used the values for the cells will be evolve incorrectly.

	x-2	x-1	x	x+1	x+2
value(t)	0.1	0.2	0.4	0.2	0.1
value(t+1)	0.12	0.14	0.56	0.14	0.12

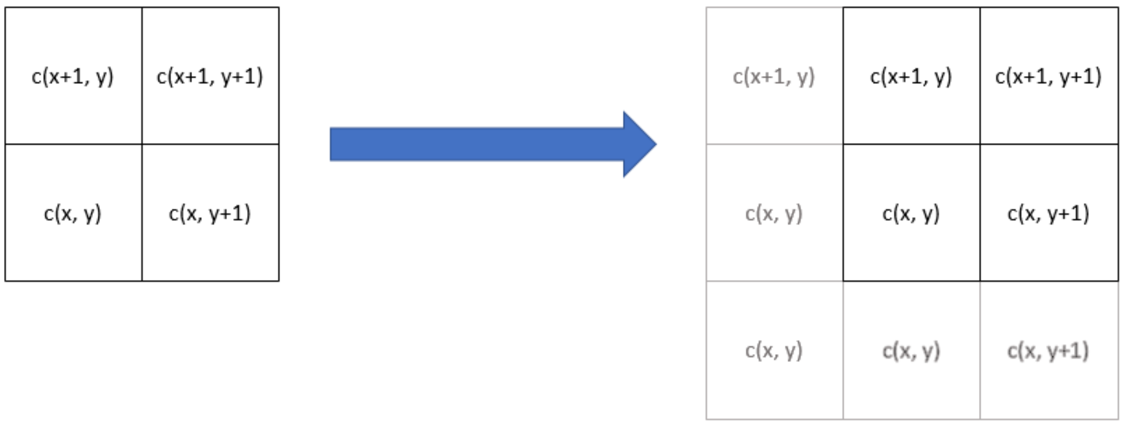
value(t+2)	0.12	0.16	0.62	0.16	0.12
value(t+3)	0.13	0.17	0.68	0.17	0.13

Table 2 Transition function example

The value of cell $c_{(x)}$ will increase over time and the probability that an object is at this place will exceed 100%. To prevent this behavior, it is necessary to normalize the probability of the function. When the probability is normalized, the formula will change to following formula:

$$pc_{k+1}(c) = \frac{\sum_{c' \in C} d(c', c) pc_k(c')}{\sum_{c' \in C} d(c', c)} \tag{3.35}$$

When a cell is on a border and has not an adjacent cell in every direction it is assumed, that the missing adjacent cell have the same rule and value



$$y \geq 0 \land y \leq \frac{lng_{(eb_{(la)})} - lng_{(wb_{(la)})}}{pix_{(a)}} \land la \subseteq a \tag{3.36}$$

$$x \geq 0 \land x \leq \frac{lat_{(nb_{(la)})} - lat_{(sb_{(la)})}}{pix_{(a)}} \land la \subseteq a \tag{3.37}$$

Where la is the area of the logic rule from c.

The area is observed by sensors. These sensors could be drones which explore the area. The sensors send their signals to the database. Each entry has a specific value, time, sensor range and a specific latitude and longitude. Based on this latitude, longitude, range and value the values of the cells will be updated.

$$sensor\ on\ position\ z\ at\ time\ k := z_{(lat,lng,k)} \tag{3.38}$$

$$range\ of\ sensor := r_{(z)} \tag{3.39}$$

$$y\ position\ of\ cell\ c := y_{(c)} \tag{3.40}$$

$$z_{(lat,lng,k)} \in \mathbb{Q} \mid z_{(lat,lng,k)} \geq 0 \land z_{(lat,lng,k)} \leq 1 \tag{3.41}$$

To find the current x and y position based on the grid of a sensor z for an area a the following calculation is made.

$$x_{(z)} = \left\lfloor \frac{lat_{(z)} - lat_{(sb_{(a)})}}{pix_{(a)}} \right\rfloor \tag{3.42}$$

$$y_{(z)} = \left\lfloor \frac{lng_{(z)} - lng_{(sb_{(a)})}}{pix_{(a)}} \right\rfloor \tag{3.43}$$

When the current grid position is found, the grid based boundaries for the sensor are calculated. For this boundary, the current position is modified by a factor which take the range and the pixelation into account. For this a distance calculation based on the Haversine formula is made.

$$dNS_{(z)} = d_{(z_{(lat)}, z_{(lng)}, (z_{(lat)} + pix_{(a)}), z_{(lng)})} \tag{3.44}$$

$$dEW_{(z)} = d_{(z_{(lat)}, z_{(lng)}, z_{(lat)}, (z_{(lng)} + pix_{(a)}))} \tag{3.45}$$

$$xLb_{(z)} = \left\lfloor \frac{\left(lat_{(z)} - \frac{r_{(z)}}{dNS_{(z)} * pix_{(a)}}\right) - lat_{(sb_{(a)})}}{pix_{(a)}} \right\rfloor \tag{3.46}$$

$$xUb_{(z)} = \left\lfloor \frac{\left(lat_{(z)} + \frac{r_{(z)}}{dNS_{(z)} * pix_{(a)}}\right) - lat_{(sb_{(a)})}}{pix_{(a)}} \right\rfloor \tag{3.47}$$

$$yLb_{(z)} = \left\lfloor \frac{\left(lng_{(z)} - \frac{r_{(z)}}{dEW_{(z)} * pix_{(a)}}\right) - lng_{(sb_{(a)})}}{pix_{(a)}} \right\rfloor \tag{3.48}$$

$$yUb_{(z)} = \left\lfloor \frac{\left(lng_{(z)} + \frac{r_{(z)}}{dEW_{(z)} * pix_{(a)}}\right) - lng_{(sb_{(a)})}}{pix_{(a)}} \right\rfloor \tag{3.49}$$

A value to compare the maximum distance of a sensor more easily is calculated based on the bounds.

$$mD_{(z)} = \left(\frac{\left(x_{(z)} - xLb_{(z)} + xUb_{(z)} - x_{(z)}\right) * dNS + (y_{(z)} - yLb_{(z)} + yUb_{(z)} - y_{(z)}) * dEW}{4}\right)^2 \tag{3.50}$$

Now a function iterates through the neighborhood from z and calculates the distance of an adjacent cell to the position of z. This distance calculation is based on the Pythagorean theorem.

Pythagoras is used here, because usually the range of a sensor is normally not that large, and the error of Pythagoras compared to the Haversine formula is small, when the distances are small and the current position is not close to the celestial poles [88]. When the area of interest is close to a pole or the sensor range is very big, the user should check weather this error could cause problems. The reason why a function is based on Pythagoras is that this calculation needs to be done frequently. And a more accurate approach needs more calculation power and the benefit is usually not very large.

The approximate number of needed calculations per minute for an area a with sensors Z in this area can be estimated with following formula:

$$\sum_{i=0}^{|Z|} \left(2 * \frac{r_{(z_{(i)})}}{\frac{distNS_{(z_{(i)})} + distEW_{(z_{(i)})}}{2}}\right)^2 * upm_{(z_{(i)})} \tag{3.51}$$

Where $upm_{(z)}$ is the number of updates per minute for a sensor z.

To check if an adjacent cell c is in range of a sensor z, the distance of this cell needs to be smaller than the $mD_{(z)}$

$$f^{i,j}(x,y): \left(x_{(c)} * distNS_{(z)}\right)^2 + (y_{(c)} * distEW_{(z)})^2 \leq mD_{(z)} \tag{3.52}$$

$$i \in \mathbb{N}_0 \land i > xLb_{(z)} \land i < xUb_{(z)} \tag{3.53}$$

$$j \in \mathbb{N}_0 \land j > yLb_{(z)} \land j < yUb_{(z)} \tag{3.54}$$

$$x_{(c)} = i \wedge y_{(c)} = j \tag{3.55}$$

For user visualization, it is necessary to create rectangles which are based on the area of the map which is shown in a browser, and the current zoom factor of the map. It is also necessary that these rectangles stay in the same position on the map, when the map is shifted around. To find the borders of each rectangle rec the following formula is used:

$$v_{sb_{(a)}} = \left\lfloor \frac{sb_{(a)}}{pix_{(a)}} - \frac{sb_{(a)}}{pix_{(a)}} \% \left(2^{20-zf} * df\right) \right\rfloor * pix_{(a)} \tag{3.56}$$

$$v_{nb_{(a)}} = \left\lfloor \frac{nb_{(a)}}{pix_{(a)}} - \frac{nb_{(a)}}{pix_{(a)}} \% \left(2^{20-zf} * df\right) \right\rfloor * pix_{(a)} \tag{3.57}$$

$$v_{wb_{(a)}} = \left\lfloor \frac{wb_{(a)}}{pix_{(a)}} - \frac{wb_{(a)}}{pix_{(a)}} \% \left(2^{20-zf} * df\right) \right\rfloor * pix_{(a)} \tag{3.58}$$

$$v_{eb_{(a)}} = \left\lfloor \frac{eb_{(a)}}{pix_{(a)}} - \frac{eb_{(a)}}{pix_{(a)}} \% \left(2^{20-zf} * df\right) \right\rfloor * pix_{(a)} \tag{3.59}$$

Where zf is the current zoom factor of google maps [89], df is the draw factor which the user can set in the tool and a is the area of interest.

After the borders for the visualization are calculated, it will be determined how many rectangles are to be drawn. To do this it is necessary to know the number of rectangles in south-north and in east-west direction.

$$num_{(sn)} \approx \frac{v_{nb_{(a)}} - v_{sb_{(a)}}}{pix_{(a)}} \tag{3.60}$$

$$num_{(ew)} \approx \frac{v_{nb_{(a)}} - v_{sb_{(a)}}}{pix_{(a)}} \tag{3.61}$$

When the numbers are known it is possible to iterate over these numbers to set the borders for each rectangle.

$$sb_{(rec_{(i,j)})} = \sum_{i=0}^{num_{(sn)}} v_{sb_{(a)}} + i * 2^{20-zf} * df * pix_{(a)} \tag{3.62}$$

$$nb_{(rec_{(i,j)})} = sb_{(rec_{(i,j)})} + 2^{20-zf} * df * pix_{(a)} \tag{3.63}$$

$$wb_{(rec_{(i,j)})} = \sum_{j=0}^{num_{(ew)}} v_{wb_{(a)}} + i * 2^{20-zf} * df * pix_{(a)} \tag{3.64}$$

$$eb_{(rec_{(i,j)})} = wb_{(rec_{(i,j)})} + 2^{20-zf} * df * pix_{(a)} \tag{3.65}$$

With the borders for each rectangle it is possible to calculate the x and y position of the cell representation. With these x and y values the cells can be easily accessed. The script will find the cell with the greatest value from the range of cells that are within the rectangle.

$$xLb_{(rec)} \approx \frac{sb_{(rec)} - sb_{(a)}}{pix_{(a)}} \tag{3.66}$$

$$xUb_{(rec)} \approx \frac{nb_{(rec)} - sb_{(a)}}{pix_{(a)}} \tag{3.67}$$

$$yLb_{(rec)} \approx \frac{wb_{(rec)} - wb_{(a)}}{pix_{(a)}} \tag{3.68}$$

$$yUb_{(rec)} \approx \frac{eb_{(rec)} - wb_{(a)}}{pix_{(a)}} \tag{3.69}$$

Summing up, a user can define an observation area. The observation area will have a certain number of cells, based on the settings. Each cell has a specific value and a specific location. The user can initialize areas of interest or cells with a specific value and can set simple logic behaviors. The cells will be updated after a specific timestamp, which is based on the logic rule, and changes their values. The value of a cell can also be changed by a sensor which search within a cell. The user can monitor the area that has been created and can observe how it evolves over time.

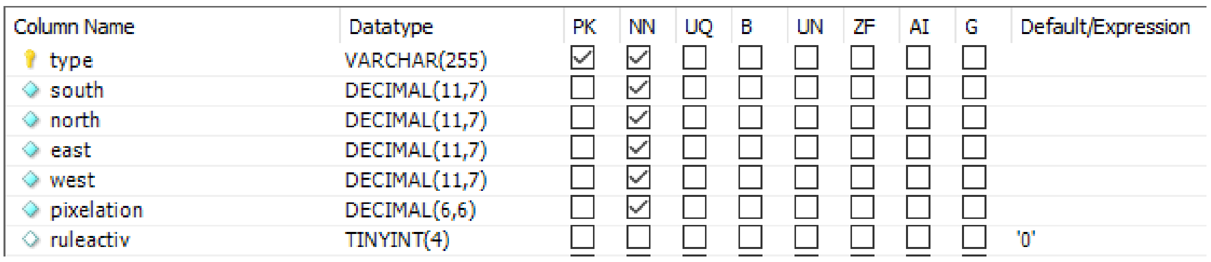
3.4 Implementation of the tool

Here the implementation of the tool is discussed in detail. First the database will be explained, because it is the basis for every other part of the programming. After the database, the server will be looked at. In the server, the main part of the model is implemented. Then the config files will be discussed in detail. In these files, the user can set up everything he needs to use the model. In the end, the visualization file is explained.

3.4.1 Database

In this chapter, the implementation will be discussed. In the beginning, the database is explained in detail. After that a closer look at the config files will be taken. Then the server will be discussed in detail and in the end the visualization will be explained.

For the database MySQL is used. In the database are several tables. Two tables are necessary for the tool. The names of these tables are *‘mapsettings’* and *‘dronelist’*. Some basic information for each area of interest is saved in *‘mapsettings’*. Each entry in this list has seven columns (figure 14).



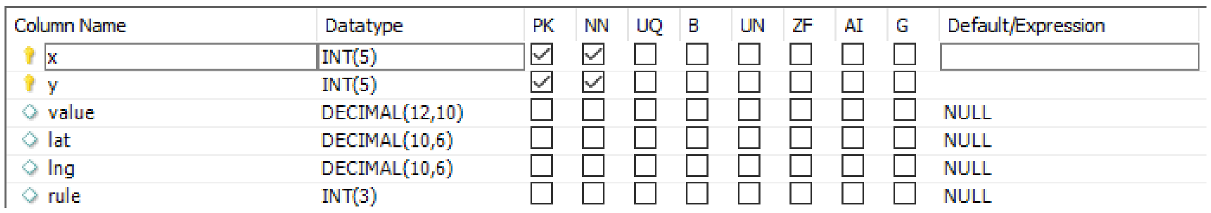
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
type	VARCHAR(255)	✓	✓	☐	☐	☐	☐	☐	☐	
south	DECIMAL(11,7)	☐	✓	☐	☐	☐	☐	☐	☐	
north	DECIMAL(11,7)	☐	✓	☐	☐	☐	☐	☐	☐	
east	DECIMAL(11,7)	☐	✓	☐	☐	☐	☐	☐	☐	
west	DECIMAL(11,7)	☐	✓	☐	☐	☐	☐	☐	☐	
pixelation	DECIMAL(6,6)	☐	✓	☐	☐	☐	☐	☐	☐	
ruleactiv	TINYINT(4)	☐	☐	☐	☐	☐	☐	☐	☐	'0'

Figure 14 Database mapsettings

The *‘type’* represents the name of the area of interest. The next four values describe its boundaries. The *‘pixelation’* was already explained and the column *‘ruleactiv’* shows if the server should apply the rules. The ‘type’ is the primary key. This will prevent from creating several areas of interest with the same name.

When a new area of interest is created two more tables are created in the database. The first table has the name of the area and contains the data for every cell, and the second table contains the data for the rules for this specific area of interest. The name of the second table is the name and a suffix.

When the user names an area of interest such as *‘testarea’*, a table with this name will be created and a second table with the name *‘testarearule’* will be created, also.

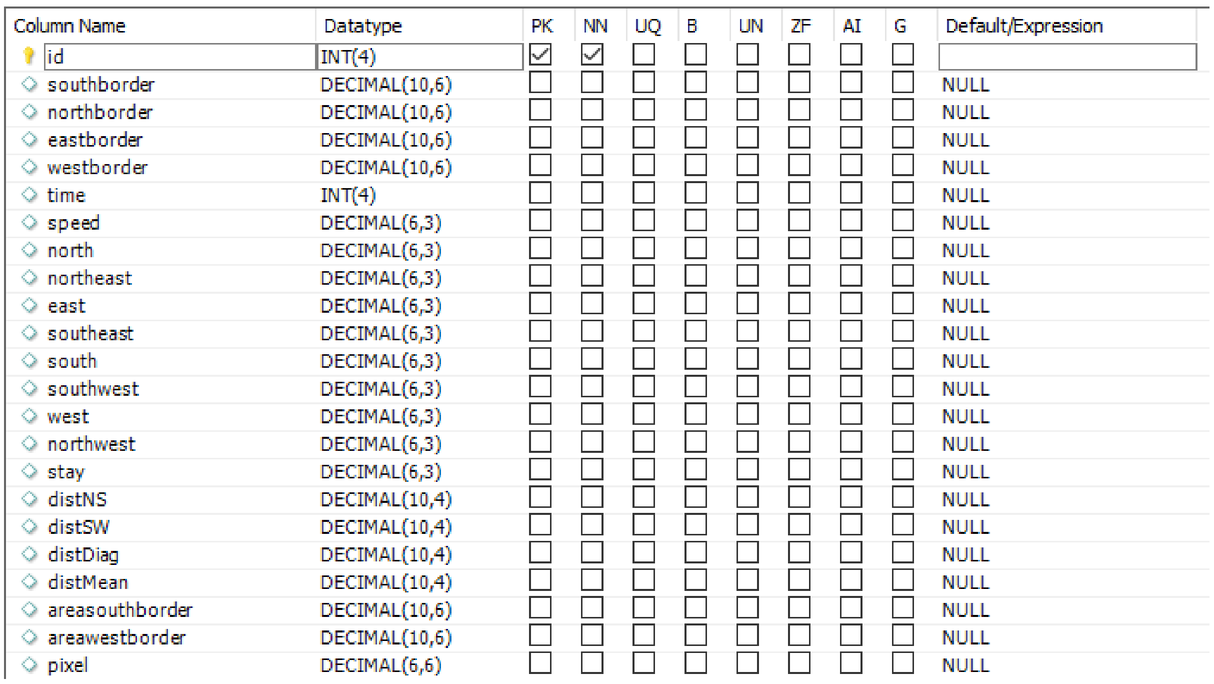


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
x	INT(5)	☑	☑	☐	☐	☐	☐	☐	☐	
y	INT(5)	☑	☑	☐	☐	☐	☐	☐	☐	
value	DECIMAL(12,10)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
lat	DECIMAL(10,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
lng	DECIMAL(10,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
rule	INT(3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Figure 15 Database area

The table which contains the data for the cells has six columns (figure 15). The first two columns represent the x and y position of the cell. The third column contains the current value. The next two columns save the latitude and longitude position of this cell. Thereby only the latitude and longitude values of the southeast corner are saved in the database.

The last column shows which logical rule is used for this specific cell. Each cell can only have one rule and when a new rule is applied to a cell the old one will be overwritten. The first two columns build a composite primary key.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(4)	☑	☑	☐	☐	☐	☐	☐	☐	
southborder	DECIMAL(10,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
northborder	DECIMAL(10,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
eastborder	DECIMAL(10,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
westborder	DECIMAL(10,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
time	INT(4)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
speed	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
north	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
northeast	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
east	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
southeast	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
south	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
southwest	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
west	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
northwest	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
stay	DECIMAL(6,3)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
distNS	DECIMAL(10,4)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
distSW	DECIMAL(10,4)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
distDiag	DECIMAL(10,4)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
distMean	DECIMAL(10,4)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
areasouthborder	DECIMAL(10,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
areawestborder	DECIMAL(10,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
pixel	DECIMAL(6,6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Figure 16 Database rule

The logical behavior for an area of interest is stored in a specific rule table (figure 16). These tables have twenty-three columns. The first entry describes the identification of the rule. This entry is also the primary key and is used to allocate each cell of the previous table the rule that is in use.

The next four columns represent the area where this rule is used. This area is always a subarea of the area of interest.

An item of interest can move with a particular velocity. This is represented as *'speed'* and the physical quantity is $\frac{m}{s}$. It is also likely to move in a specific direction or stay at its current position. The columns *'north', 'northeast', 'east', 'southeast', 'south', 'southwest', 'west', 'northwest'* and *'stay'* represent this.

The mean distance between two cells for the area where the rule is applied, and the pixelation are also stored in order to make faster calculations by avoiding unnecessarily accessing to the database and do the same calculations several times. The distance between two cells in a north-south direction is stored in *'distNS', 'distEW' this* saves the distance between two adjacent cells in east-west direction, and *'distDiag'* stores the distance between two diagonal adjacent cells. The mean of the three distances is stored in *'distMean'*. In *'pixel'* the pixelation of the area of interest is secured. When the area where the rule is applied is very large, this can cause some problems because the real distance between two adjacent cells where the rule is applied is not the same as the saved values. Usually this error can be ignored. It only occurs when calculations with the latitude values are done. When one cell has a pixelation of 0.0001 and the difference between the north and the south boundary of the area is 0.01 the error is about 1mm. An area of this size has roughly a south-north extension of 1 km and one cell has roughly an extension of 10m. When the cell pixelation is 0.1 and the difference of the area boundaries is 10 the error would be roughly 14m. The size of a cell would be about 10 km and the area of interest would be around 1000km. These errors will increase when the area of interest gets closer to the earth's poles.

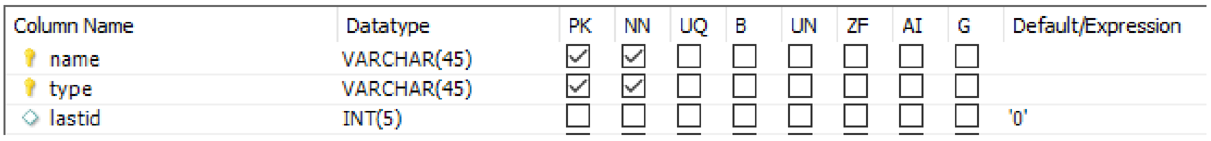
When the rule is applied it is also necessary to know the south and the west boundaries of the area of interest to calculate the correct x and y position of a cell which is affected by the rule. These values are stored in *'areasouthborder'* and *'areawestborder'*.

The *'time'* field shows how often the rule is executed in server time. This value is based on several other values and is calculated to have as few updates as possible without losing accuracy. It tries to update the cells every time that the item of interest moves to the next cell based on the speed of the item, the pixelation of the cell and their latitude and longitude values.

$$upd = \frac{\frac{\sum_{j=y-1}^{j=y+1} \sum_{i=x-1}^{i=x+1} d(lat, lat + pixel * i, lng, lng + pixel * j)}{8}}{speed} \tag{3.70}$$

Where $d(x, x', y, y')$ is the distance based on the haversine formula.

$$lat = \frac{southborder + northborder}{2} \text{ and } lng = \frac{eastborder + westborder}{2} \tag{3.71}$$



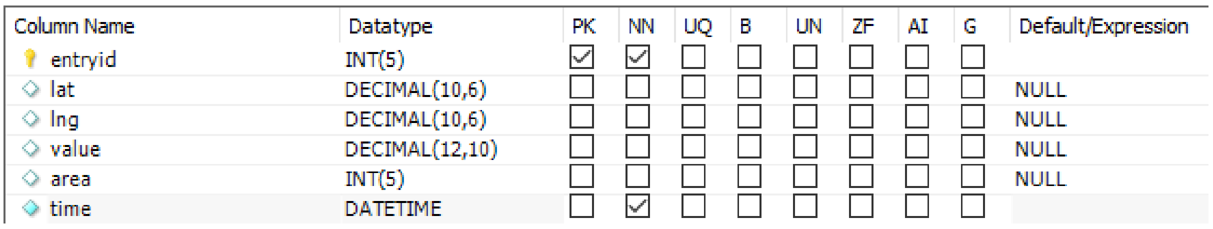
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
name	VARCHAR(45)	✓	✓							
type	VARCHAR(45)	✓	✓							
lastid	INT(5)									'0'

Figure 17 Database dronelist

The table *'dronelist'* contains the information for each sensor (figure 17). In this list, the name and type of the sensor are stored. The *'type'* describes which area of interest is affected by this sensor. The value in *'type'* must match the name of an area of interest. The *'lastid'* is the last entry which was processed by the server. When a sensor has made several new entries, and is not able to send this data to the server at the time it makes the observation, it can send all collected data to the server at once and the server will still evaluate this data correctly.

When a sensor can affect several areas of interest it is easily possible to build the desired behavior. To do so, several entries with the same name of the sensor need to be made. Each entry has another value for the *'type'* but the same value for *'name'*.

When an entry with a new name is created, a new table with the name of the sensor will be created. To avoid conflicts between areas of interest and sensors it is necessary not to use the same name for a sensor and for an area of interest.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
entryid	INT(5)	✓	✓							
lat	DECIMAL(10,6)									NULL
lng	DECIMAL(10,6)									NULL
value	DECIMAL(12,10)									NULL
area	INT(5)									NULL
time	DATETIME		✓							

Figure 18 Database sensor

When a new sensor is added to *'dronelist'* a table for this sensor is created (figure 18). This table contains five columns. The first two columns describe its current position based on latitude and longitude values. The next value is the likelihood that the sensor has found an item of interest. *'Area'* describes the range of the sensor when it made the last observation, and is measured in meters. The last value is the time when the last observation was made.

3.4.2 Server

The server is written in NodeJS. The server fulfills several tasks. It is responsible to navigate the user correctly through the config menu, to receive the user input through the config files, to responses requests of user. The server manages the updating of each area of interest. To do this it handles the input of the sensors and handles the logical rules for every area of interest.

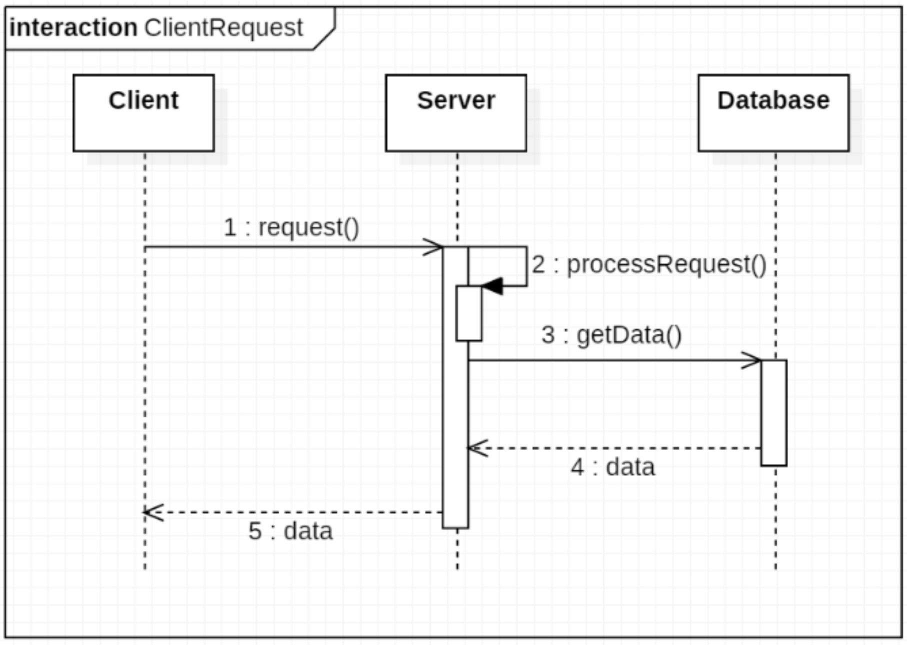


Figure 19 Sequence client update

In the first section, the server handles the user requests (figure 19). The user sends XMLHTTPRequests to the server. This is an interface to transfer data with an HTTP-Protocol [90]. The server deserializes the request and can read out the information that the user sent. These requests have usually two parts. The first part describes what the user wants. The second part may contain additional information. It is possible that the second part does not contain any data. This depends on the request type.

The server also has an update loop (figure 20). This loop is called every second. In the beginning, the Loop checks for new tables in the database. Within this function an asynchronous call is made. The answer of this call will come after the update function is finished, so a change in the table structure of the database will not be applied in this function call. After this it will be checked if a sensor has made a new entry. When a new entry is made, the area of interest will be changed according to the sensor data. This function also has an asynchronous call. Within this function the server basically checks if a new entry is in the sensor's data table. Then the update loop will iterate over every area of interest and check if the logical behavior is active. When it is active, the logical behavior will be applied to the area.

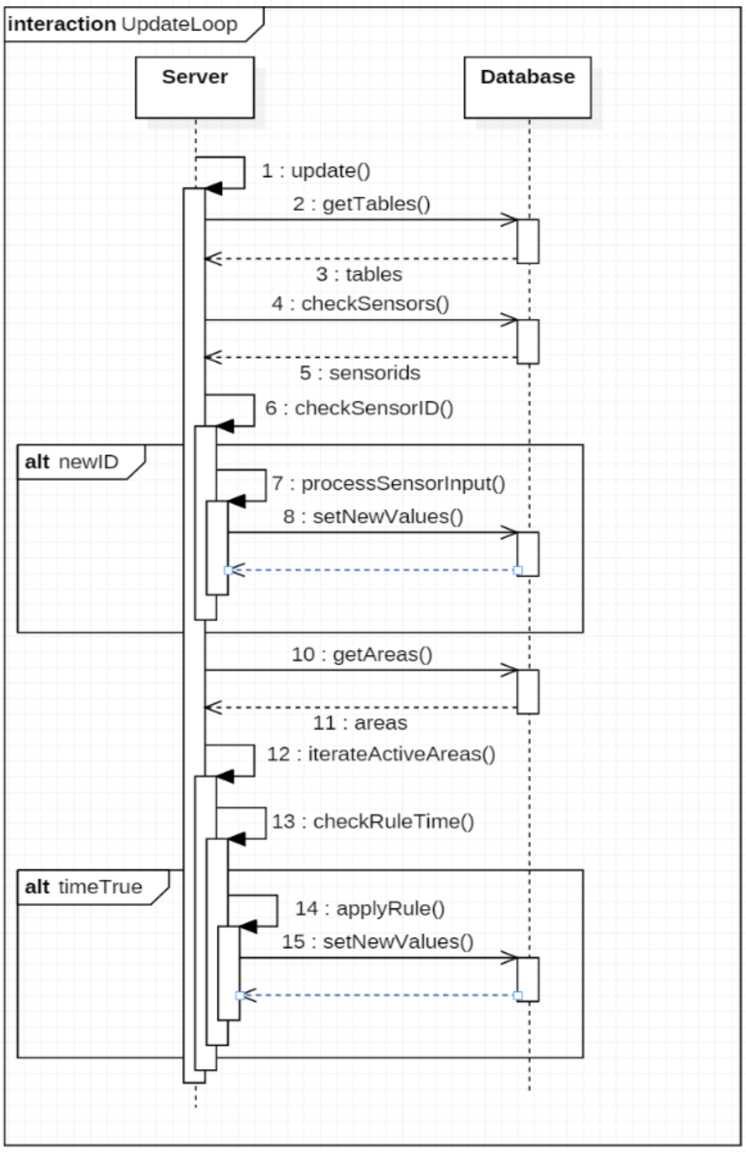


Figure 20 Sequence update loop

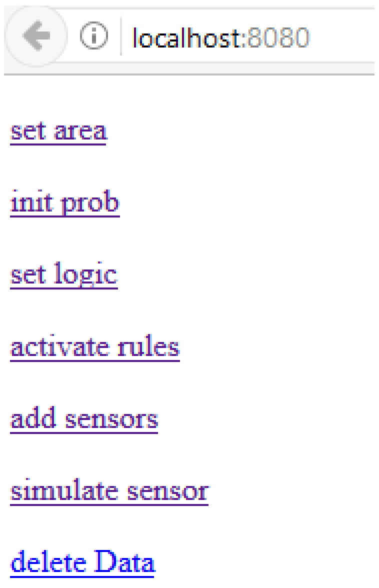
3.4.3 Config files

In the config files, the user has the ability to set up the observation scenario. As shown in chapter 3.2, the user has various options to define the area of interest, initialize this area and create simple logical rules for it. He can also define which sensor infects which area of interest.

In the config files, the user interacts several times with the Google Maps API. This API provides an easy to use environment to display a geographical map [92]. On this map, the user can take several actions, depending on the currently loaded file.

To load the config files the user can navigate through an index file (figure 21). When the user accesses the server, this file will be loaded. Within this file the user can select each separate config file.

Whenever the user has done something in these files, the files will send a request to the server. The server will process those requests.


localhost:8080

set area

init prob

set logic

activate rules

add sensors

simulate sensor

delete Data

Figure 21 Index file

3.4.4 Sensor data

A sensor can send its data to a server. When it does so, it is necessary that the sensor is assigned to an area of interest. It is possible that the sensor could be assigned to several search areas, but at least one assignment is necessary.

Whenever the sensor collects data it can send it to the server. The data has to be a string which contains several parts of information. It needs to send a prefix, *"sensorData"*, the name of the sensor, its current position in latitude and longitude values, the ranges of the sensor and the value if it detects something. Each entry is separated by a semi-colon. A string which is sent to the server might look like this: *"sensorData;sensor1;64.578;-121.365;10;0.95".*

The range of the sensor is a number, and describes the radius of the detection area in metres. The value is a rational number between zero and one. A value of zero means, that the sensor has not found anything. A value of one means that the sensor has detected something. When the sensor has a glimpse probability, this glimpse probability needs to be accounted before the value is sent to the server.

Usually, a sensor either can detect an item of interest or cannot detect it. This can be described as:

$$z_{(lat,lng,k)} = \begin{cases} 1, if\ item\ of\ interest\ is\ detected \\ 0, otherwise \end{cases} \tag{3.72}$$

For a sensor z at the position (lat, lng) at time k.

The glimpse probability $gp_{(z_{(lat,lng,k)})}$ describes the chance, that the detection is correct. For a sensor z at the position (lat, lng) at time k.

In the mathematical model, these two steps are described with only one formula, 3.41, because for the model the technical details of the sensors are irrelevant.

3.4.5 Visualization

The user can load every area of interest that is stored in the database (figure 22). It is necessary to reload the visualization page, when an area of interest has been added after the last reload. The reason for this is that the visualization page will load the data of every area of interest when it is loaded. After that it will not load this data again. Usually it is unlikely that a new area of interest is added while a user observes an area. This behavior was implemented to avoid unnecessary request

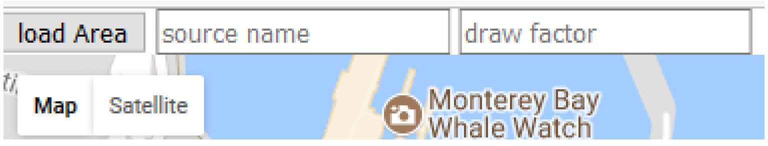


Figure 22 Load area visualization

The update loop is called every five seconds (figure 23). Within this loop the script will check if it is waiting for an older request. This is to prevent conflicts between requests, because it can happen that the handling of a request will take more time than expected due to a bad connection and the amount of data, which needs to be sent and evaluated.

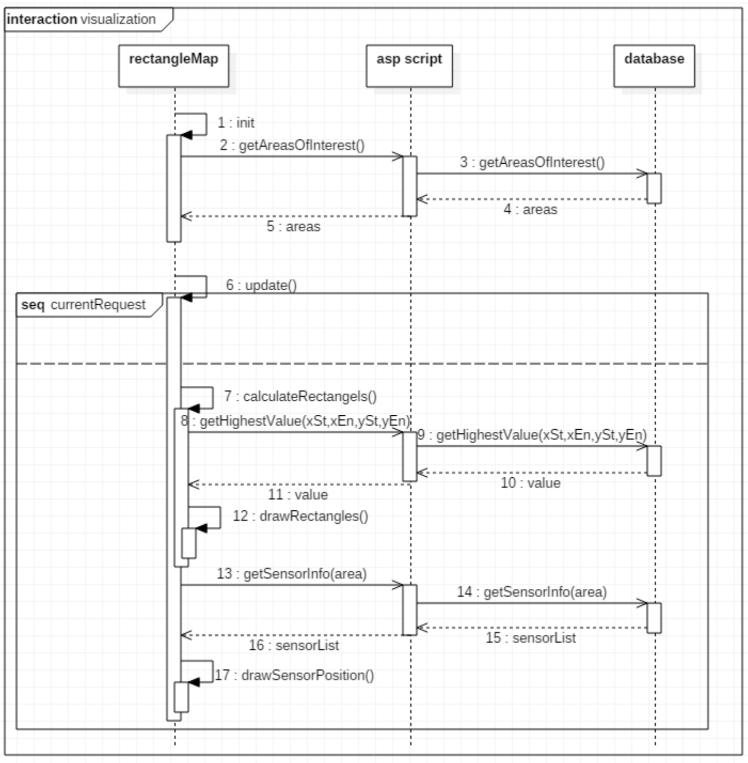


Figure 23 Sequence visualization update

When the last request is finished two requests are sent to get the specific data. The first request will get the highest value of all cells within a specific rectangle. After the data for every rectangle has been received, these rectangles will be drawn. The color of a rectangle is based on the value. If the value is zero, the color will be green, if the value is 1 the color will be red.

The second request receives the data of all sensors which affect the visualized area of interest. For each sensor, a track will be drawn, which shows how the sensor has moved in the past. The sensor current position is marked with a dot and the user can read the name of each sensor, by hovering his mouse on the dot.

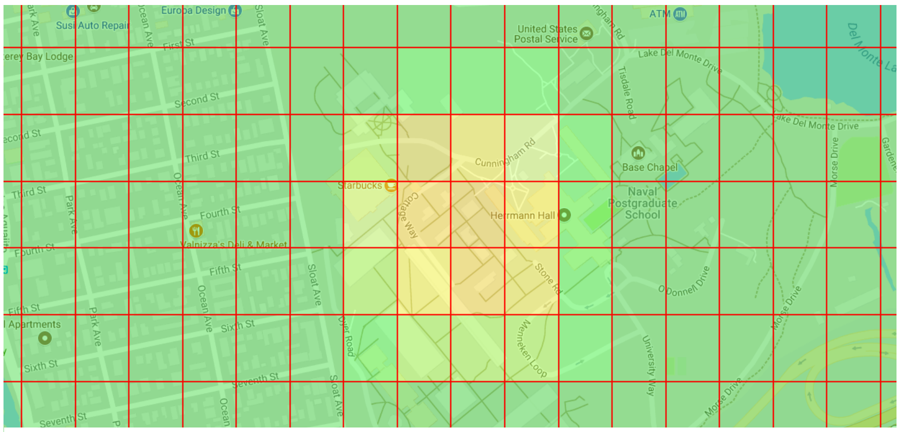


Figure 24 Visualization example init

In figure 24, the visualization of an area is shown. Most rectangles are green, which indicates, that the probability an item of interest is in this area is low. In the middle the probability is greater.

In figure 25, a sensor is added to the area of interest. The sensor found something at its position. The area, where the sensor found something, changed its color and the position of the sensor is marked as a dot.

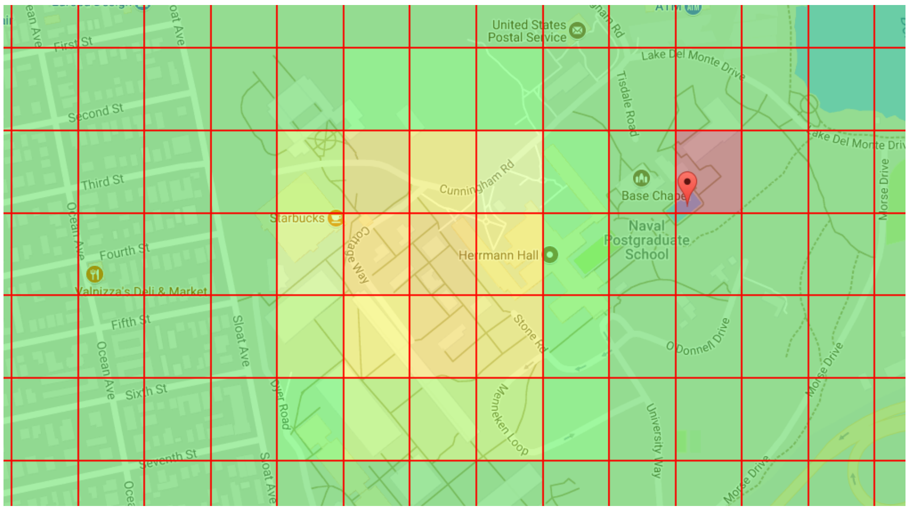


Figure 25 Visualization example sensor detects something

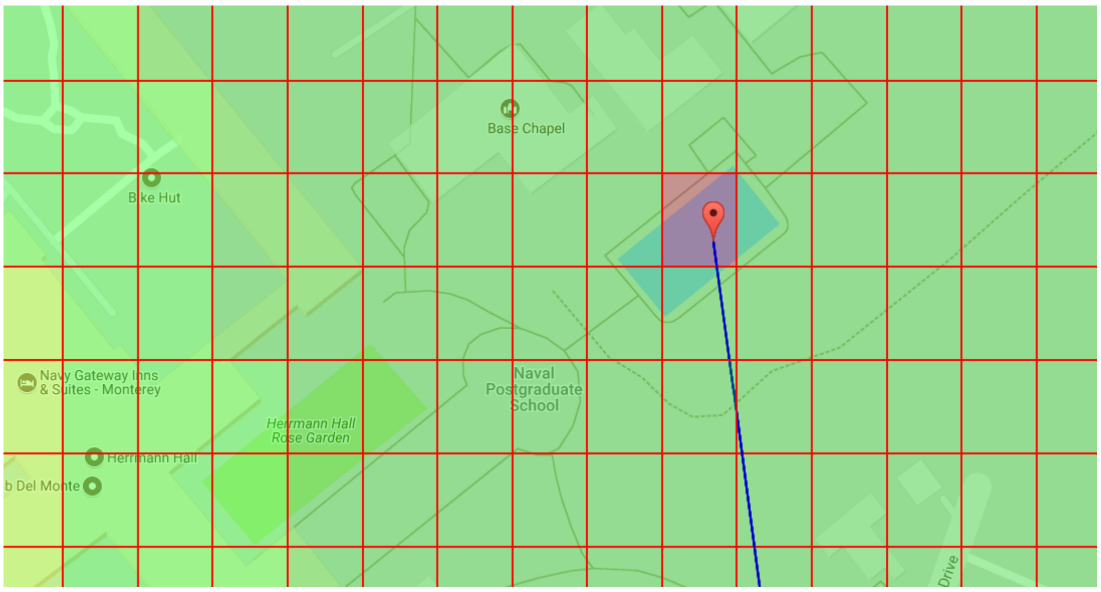

Base Chapel
Bike Hut
Navy Gateway Inns & Suites - Monterey
Herrmann Hall Rose Garden
Herrmann Hall
Naval Postgraduate School
Drive

Figure 26 Visualization example zoom

The user can zoom into the visualization to get a closer view of the area (figure 26). The rectangles get rearranged and the user can enclose the area where the item of interest probably is located.

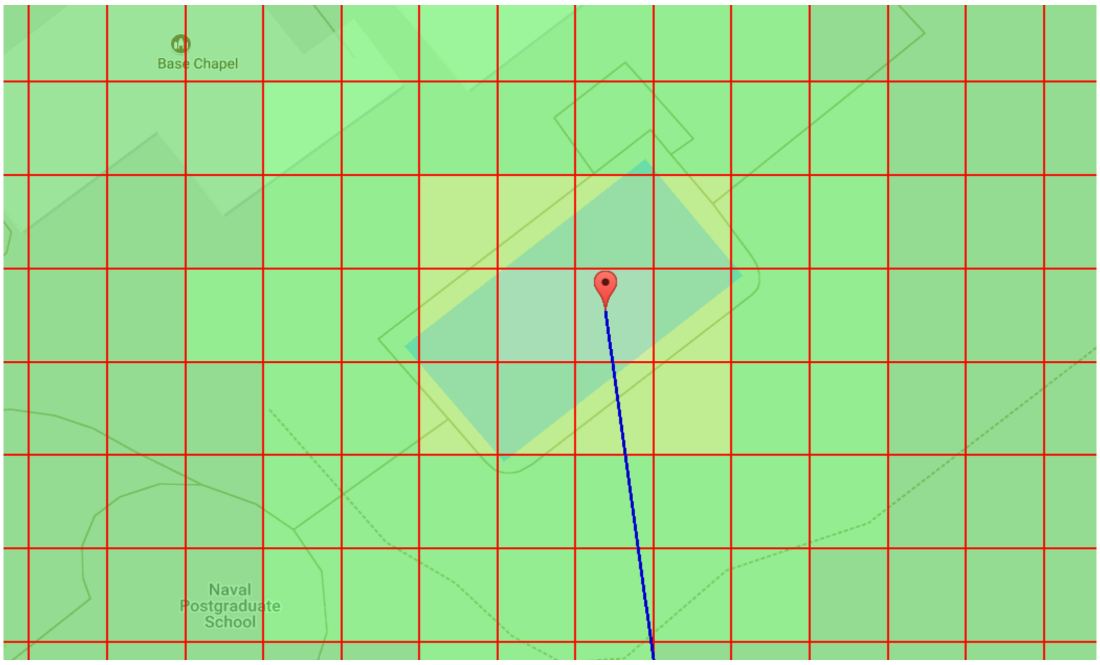


Figure 27 Visualization example update

After a certain time, the rectangles change color again (figure 27). Due to the rule for this area of interest, an item of interest can move in any direction.

4 Analysis

This section takes a deeper look into the issues and the performance of the tool. In the beginning the issues are explained. After that some benchmark tests are shown.

4.1 Issues of the tool

The model has several issues and could be improved. The main problems will be discussed in this section. Some of the issues are technical limitations, some of them are used to improve the performance of the tool and some of them are limitation of the model. The limitations of the model could be removed to make the tool more powerful but this will be time costly and it is better to do practical tests with the tool to see what will be the best way to deal with the limitations.

4.1.1 Inaccurate calculations

Several of the formulas used in the tool have a small margin of error. Usually this should not be a problem, because the error is negligible. One inaccuracy is caused by the numerical calculation. Especially in the haversine formula, a floating-point error [93] can occur. In the formula, $hav\left(\frac{d}{r}\right)$ should not exceed one. Because the distance is only a real number when $hav\left(\frac{d}{r}\right)$ is between zero and one. The function $hav\left(\frac{d}{r}\right)$ approaches one, when the target point is on the other side of the sphere. So, this error is more likely to occur when the distances are very great. It is possible to avoid this when the haversine formula is used with $\cos()$ instead of $\sin()$. But this will lead to a floating-point error for small distances. This tool usually uses small distances.

As already explained in section 3.2 another error occurs due to the fact that the earth is not a perfect sphere. In the following figures, the error between the ellipsoidal shape of the earth, without hills and valleys, is compared to a perfect sphere with the same radius as the earth. The errors are plotted between endpoints at $(lat, lng) = (mu, 0)$ and $(x, lambda)$ is a function of latitude x between -90 to 90 degrees. The rows correspond to values of mu at $\{0, 22.5, 45, 67.5\}$ degrees and the columns to values of lambda at $\{0, 45, 80, 180\}$ degrees [93]. In figure 28 the absolute error in kilometers is shown and in figure 29 the relative error is represented. The absolute error may appear to be large when only small areas of interest are observed. But this error occurs to every calculation, so everything in this area of interest has the error offset.

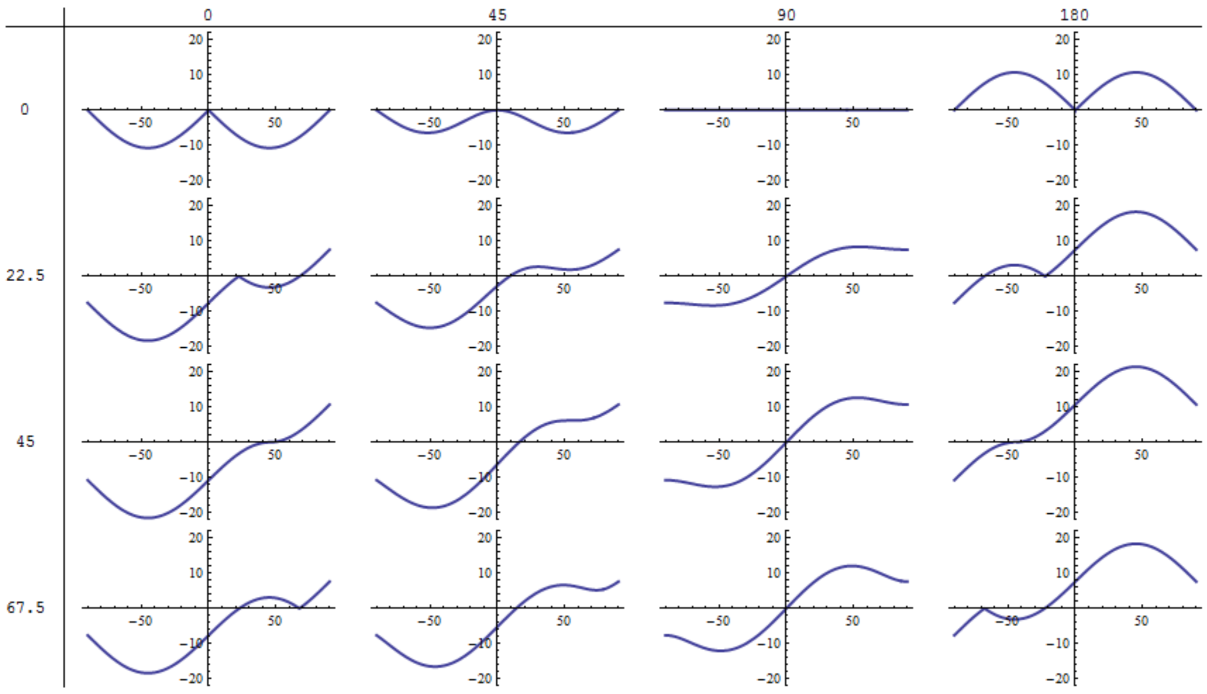


Figure 28 Absolute error of Haversine [93]

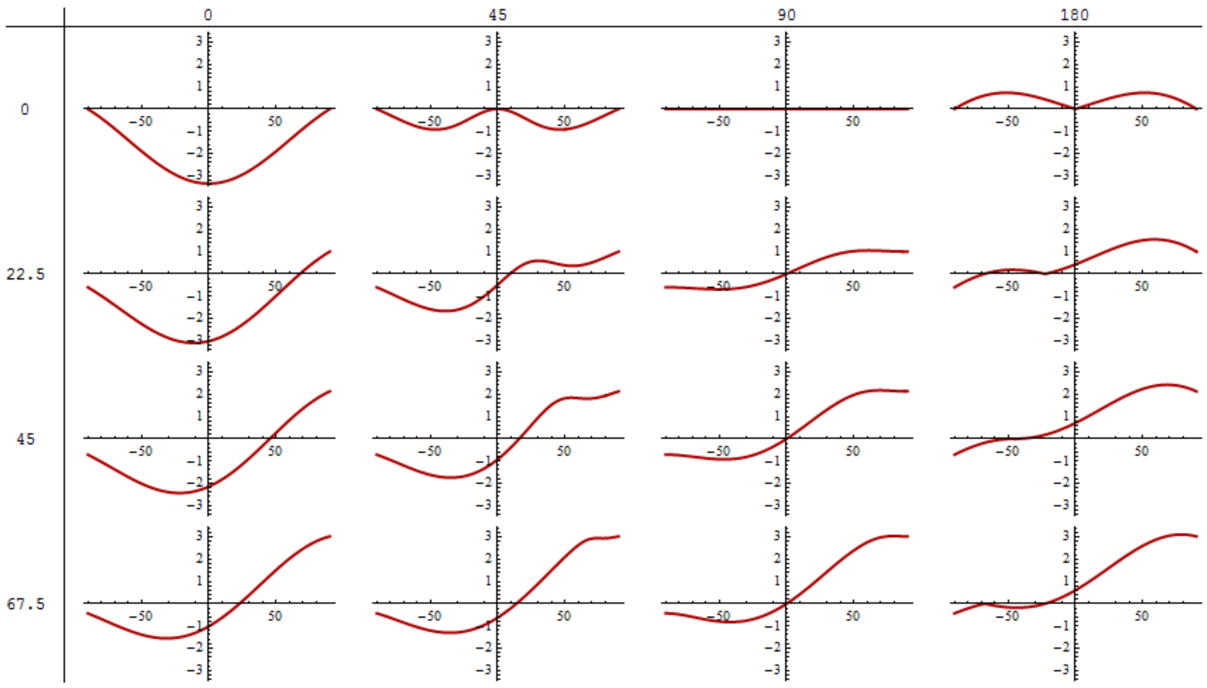


Figure 29 Relative error of Haversine [93]

Another way to visualize this error is to set a fixed endpoint and let the other point vary. In figure 30 the first point is at 45 degrees latitude and 0 degrees longitude. The error is again in absolute values. In figure 31 the same start point is used, but the error map is wrapped around a globe.

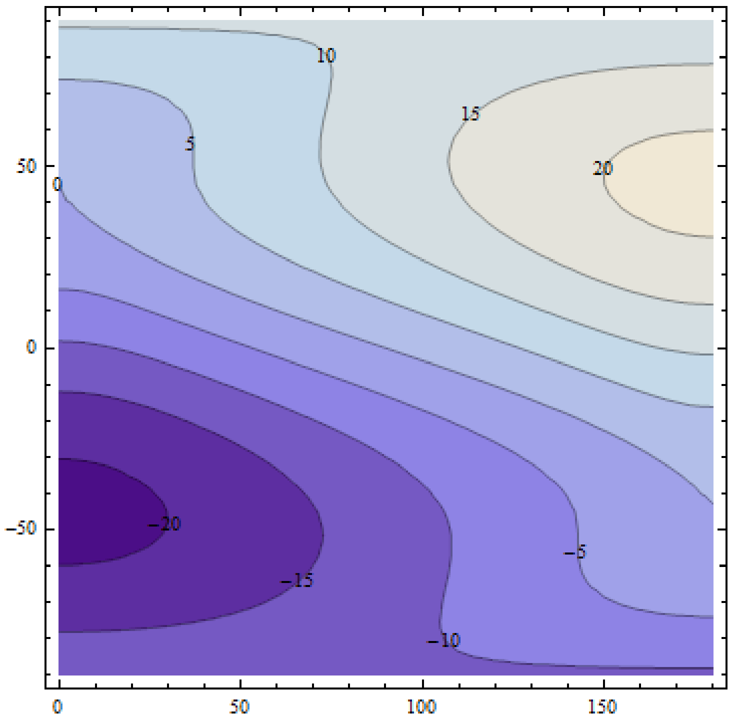


Figure 30 Absolute error of Haversine 2 [93]

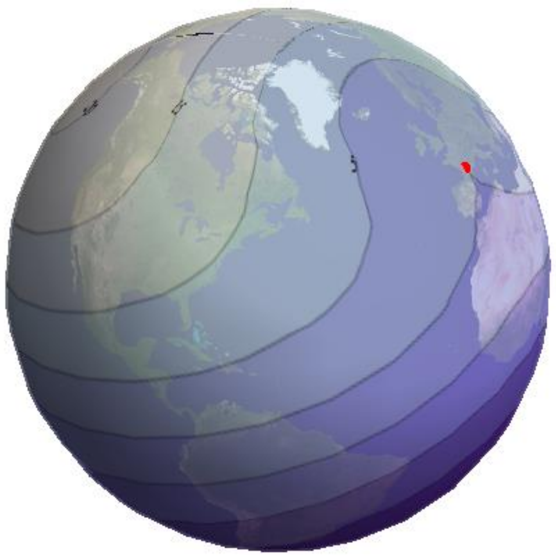
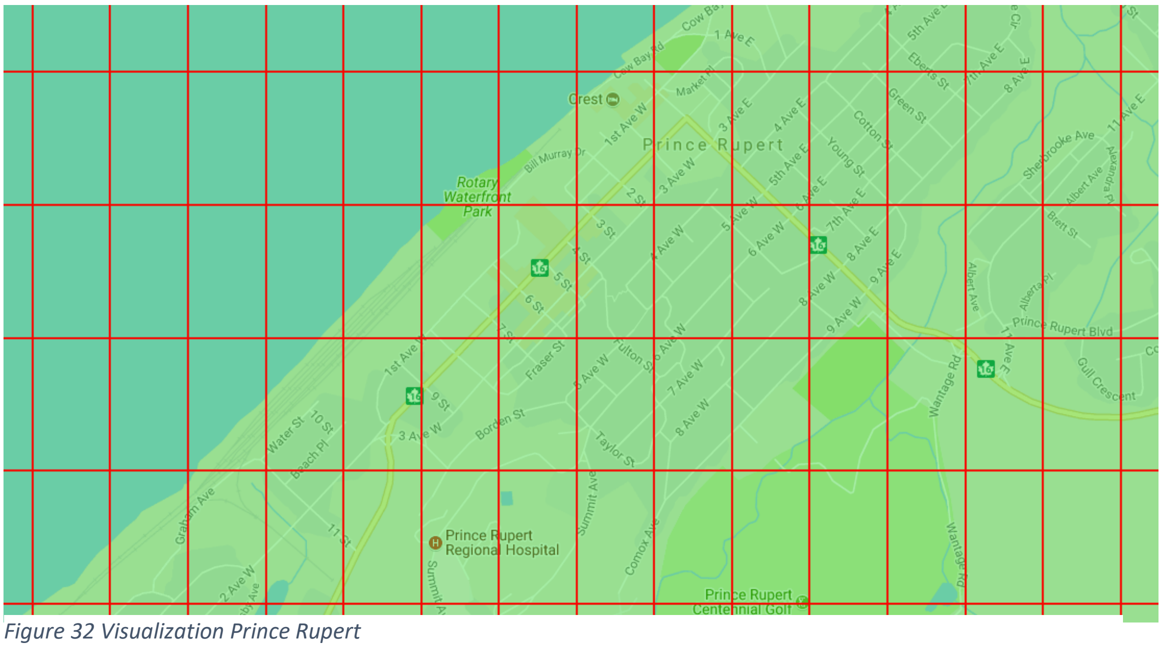


Figure 31 Absolute error of Haversine 3 [93]

4.1.2 Distorted visualization

The tool uses latitude and longitude values to store the area of interest and to represent the data within an area. The distance between two latitude values is always the same. For a minute of arc of longitude values the distance is always about 1852 meters. This is known as a nautical mile. The distance of a minute of arc for latitude values is also one nautical mile, while the distance on the latitude values δ (north and south) is up to $\cos(\delta)$ smaller than one nautical mile. This will lead to a distorted visualization, when the area of interest is near the poles. Due to this error, it may be harder to understand the representation properly. But even with this representation error the calculations in the background have only the already explained calculation error.



Prince Rupert
Rotary Waterfront Park
Prince Rupert Regional Hospital
Prince Rupert Centennial Golf

Figure 32 Visualization Prince Rupert

In figure 32 an area of interest is in Prince Rupert (54.312, -130.322) and in figure 33 another area of interest is in Manzanillo, Costa Rica (9.694, -85.203). In both figures the angular distance between the north-south and east-west lines is the same.

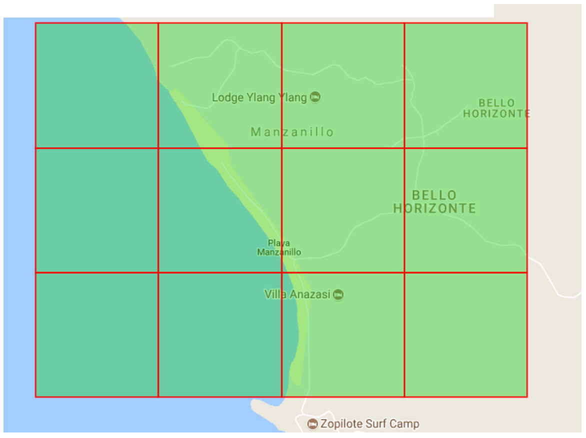


Figure 33 Visualization Manzanillo

4.1.3 Simple logic

The logical behavior that a user can set for an area of interest is limited. It is hard to create complex behavior. It is possible to describe water flows and ocean currents for search and rescue missions on sea, or to implement the approximate behavior of the moving pattern of a moving item of interest. The user can create rules for every street such that a car can move on the street in a certain direction and is not able to move within buildings. It is also possible to change the speed of the car for a specific street or area. But it is not possible to create a behavior pattern for items of interest. That depends on the search mission. For example, it is not possible for an item of interest to try to avoid sensors.

When rules with different movement speeds share a same border, a bias can occur. The update time for a rule is based on the approximated speed of an item of interest within the rule area. This is done to reduce the required number of calculations. But when the speed of one rule $r_{(1)}$ is double the amount of the speed of another rule $r_{(2)}$ and there are adjacent cells $c_{(1)}$ and $c_{(2)}$, where the rule for $c_{(1)}$ is $r_{(1)}$ and the rule for $c_{(2)}$ is $r_{(2)}$. Then cell $c1$ will be updated twice as frequently as $c_{(2)}$. The value of $c_{(1)}$ influences the value of $c_{(2)}$ and vice versa, because they are adjacent. The different update speeds will cause an effect that the value of the cell with the lower speed has a slightly higher influence on the field with a greater value. In the following tables four scenarios are represented. In the first two scenarios, the cells $c_{(0)}$ and $c_{(1)}$ are updated every timestamp, cell $c_{(2)}$ and $c_{(3)}$ are updated every second timestamp. In the third and fourth scenario both cells get updated at the same time. In every scenario, the item of interest can move to an adjacent cell or can stay at the current cell with the same probability.

The value of a cell which is less frequently updated, has a higher impact on the overall values. Especially when many rules with different speeds and with only a small area, where this rule is used.

The impact from one cell $c_{(x)}$ an adjacent cell $c_{(y)}$ can be determined by following formula:

$$imp\left(c_{(x)}, c_{(y)}\right) = \frac{upd_{(c_{(x)})}}{upd_{(c_{(x)})} + upd_{(c_{(y)})}} \tag{4.1}$$

Where upd is the formula explained in 3.70.

Scenario 1	$c_{(0)}$	$c_{(1)}$	$c_{(2)}$	$c_{(3)}$		Scenario 2	$c_{(0)}$	$c_{(1)}$	$c_{(2)}$	$c_{(3)}$
value(t)	0.5	0.5	0	0		value(t)	0	0	0.5	0.5
value(t+2)	0.5	0.33	0	0		value(t+2)	0	0.17	0.5	0.5
value(t+3)	0.42	0.28	0.11	0		value(t+3)	0.8	0.22	0.39	0.5
value(t+4)	0.35	0.27	0.11	0		value(t+4)	0.15	0.23	0.39	0.5
value(t+5)	0.31	0.24	0.13	0.06		value(t+5)	0.19	0.26	0.37	0.44
value(t+6)	0.28	0.23	0.13	0.06		value(t+6)	0.22	0.27	0.37	0.44
value(t+7)	0.25	0.21	0.14	0.09		value(t+7)	0.25	0.29	0.36	0.41
value(t+8)	0.23	0.20	0.14	0.09		value(t+8)	0.27	0.30	0.36	0.41
value(t+9)	0.21	0.19	0.14	0.11		value(t+9)	0.29	0.31	0.36	0.39
…										
value(t+33)	0.15	0.15	0.15	0.15		value(t+33)	0.35	0.35	0.35	0.35
Scenario 3	$c_{(0)}$	$c_{(1)}$	$c_{(2)}$	$c_{(3)}$		Scenario 4	$c_{(0)}$	$c_{(1)}$	$c_{(2)}$	$c_{(3)}$
value(t)	0.5	0.5	0	0		value(t)	0	0	0.5	0.5
value(t+2)	0.5	0.33	0.17	0		value(t+2)	0	0.17	0.33	0.5
value(t+3)	0.42	0.33	0.17	0.08		value(t+3)	0.08	0.17	0.33	0.42
value(t+4)	0.38	0.31	0.19	0.13		value(t+4)	0.13	0.19	0.31	0.38
value(t+5)	0.34	0.29	0.21	0.16		value(t+5)	0.16	0.21	0.29	0.34
value(t+6)	0.32	0.28	0.22	0.18		value(t+6)	0.18	0.22	0.28	0.32
value(t+7)	0.30	0.27	0.23	0.20		value(t+7)	0.20	0.23	0.27	0.30
value(t+8)	0.29	0.27	0.23	0.21		value(t+8)	0.21	0.23	0.27	0.29
value(t+9)	0.28	0.26	0.24	0.2		value(t+9)	0.2	0.24	0.26	0.28
…										
value(t+17)	0.25	0.25	0.25	0.25		value(t+17)	0.25	0.25	0.25	0.25

Table 3 Rules with different update speed

As expected, the impact of the cell with the lower update speed is greater. When the cells have the same update speed, the impact of them is the same.

4.1.4 Limited area size

The size of an area of interest is limited due to the system where the server is running or the database. Only a certain number of entries can be sent from the database to the server or stored in the memory of the server. The actual size of an area of interest is limited to the maximum number of entries, which can be stored, and to the pixelation. And it is also limited to the size of a single rule area.

The server can only store data depending on the available computer memory. The required data for each cell can easily be calculated. Each cell contains 5 values. The first two values are integers (x and y position), the other value are floating points (probability, item of interest is within this cell and the latitude and longitude position). JavaScript stores every number as a 64-bit Floating Point [94]. So, for every cell, five 64-bit values need to be stored. These values are stored in an array, so every cell needs at least 40-bytes of memory storage and the overhead of the array. With a conservative assumption, the server can use 80% of its memory capacity to store the data. Depending on the number of areas and their rules it is necessary to have enough memory space to store every area of interest the server is currently processing. It only needs to consider areas of interest with active rules.

When a new area is created the server does not need to store all the data at once. The cells are created iteratively and when a certain number of cells have been created they will be sent to the database and the array will be cleared.

The 80% available memory needs to be divided by the number of active areas. When only one area is active the server can use the 80% memory for this area.

Usually it will not store the whole data at the same time, but to the asynchronous call functions when the server accesses the data it can happen that it must store all the data at one time. When the rules do not cover the whole area, it needs even less dataspace.

Assuming, that a server has 2 GB memory limit, it can store 80,000,000 cells at once. With this number of cells an area of interest can be approximate 8900 x 8900 cells big. Depending on the pixelation of the cells, the real size the area can cover can be calculated. With a pixelation value of 1 the latitude size of a cell is approximate 111km and the longitude size is $111km * \cos(latitude)$. In this example, the latitude value is 0°. With a pixelation of 0.0001, a cell would cover an area of 121 m^2. With the previous assumption a server with 2GB memory can store an area of interest of 9680 km^2 size.

Another limitation is the amount of data that can be transferred in one packet between the server and the database. The default packet size of MySQL is 16 MB and the largest possible size is 1GB [95]. Theoretically, with the default setting only, 400,000 cells can be sent and 25,000,000 can be send at maximum. But from time to time communication errors appear, when the number of cells is greater than 100,000 cells with the default settings. During our tests, these errors never occurred when the number of cells were smaller than 100,000.

With a pixelation of 0.0001 it is possible to cover 12.1 km^2.

This limitation only affects the size of a single rule and not the total size of an area of interest, because only the data of a rule is hard to split. The total data can be easily split.

When an area of interest is created it is also necessary that it is greater than the minimum size of a cell based on the set pixelation. It is not possible to create an area of interest where the difference between its south and north border or between its east and west border is smaller than the pixelation.

4.2 Benchmark Tests

It is necessary that the tool can do all calculations in real-time. Otherwise it is not possible to use the tool to monitor a real-time search mission. In this section, several benchmark tests are presented to show that the tool is capable to do the necessary calculations fast enough to be used as a real-time decision support tool. For these tests, an Intel Core i5-6200U with 2.3GHz and a Turbo Boost up to 2.8 GHz are used. The computer has 8 GB DDR4 memory and all the data was stored on a SSD hard drive. This PC was used as a server and a client at the same time, so time delays for communication over the internet can be neglected. As a server, the Internet Information Services from Windows were used on a Windows 10 64-bit operating system. The browser was Firefox on Version 54.0.1 and the database was MySQL Community Server with version 5.8.18. Each test was made ten times and the data presented in this section reflects the average value of each test.

In the beginning a user would set up an area of interest. To do so he can create the area of interest within the config files. For this part, it is not necessary to be in real-time because the observation will not be made when the user initializes everything relevant. The time that is needed to create the database depends on the size and the pixelation of the area. These factors determine how many cells are created. Table four shows, how long it takes to create a database for an area of interest with a certain number of cells.

Number of cells	100	500	10.000	50.000	1.000.000
Time to create database in milliseconds	27	28	286	847	18816

Table 4 Creation time of database

One of the most important formulas in the tool is the Haversine formula to make accurate distance calculations. This formula is complex and must do several trigonometric functions and a root calculation. It would be possible to calculate the distances based on Pythagoras but this would not be as accurate as the Haversine function and the time needed to make a single distance calculation can be neglected. The calculation of the Haversine function takes approximately four to five times as long as a Pythagoras functions. The time to make ten million distance calculations is shown in table 5.

Number of calculations	Haversine function	Pythagoras function
10.000.000	431 milliseconds	94 milliseconds

Table 5 Time comparison Haversine-Pythagoras

During a search mission, sensors will make detections and send them to the server. Based on the particular sensor a certain area gets searched and a specific number of cells are updated. The larger the detection radius of a sensor, the more cells that must be updated. The pixelation of the area of interest influences the size of a cell and also influence the number of cells that get updated by a sensor. The number cells that are affected by a sensor increases significantly with the detection radius as shown in figure 34. This leads to an increased update time when a sensor has a high range. The times are represented in table 6.

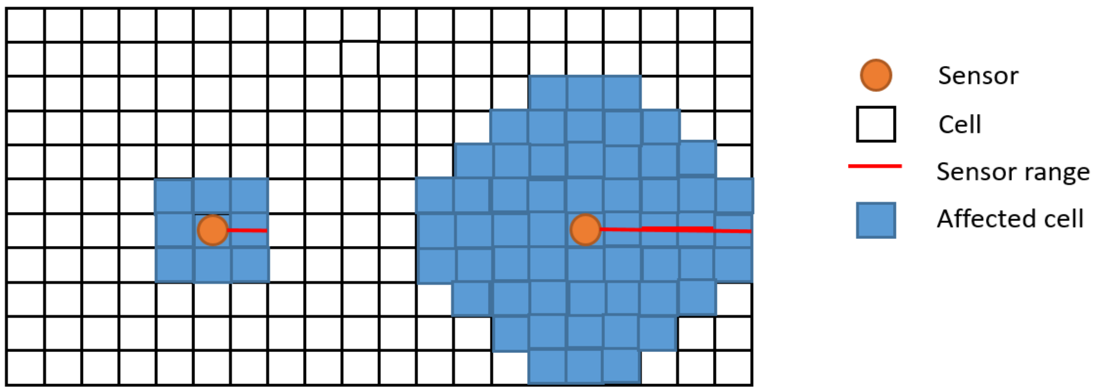


Figure 34 Cells affected by sensor depending on range

The number of cells affected by a sensor increases approximately with the square of the range in relation to the pixelation of the area of interest. It is not precisely the square because the cells are usually not a perfect square, but are a rectangle based on their latitude and longitude values and the given pixelation.

Affected cells	1	1	1	9	9	9	110	110	110
Number of sensors	1	10	20	1	10	20	1	10	20
Time to update all in milliseconds	16	55	72	31	224	412	257	2437	4828

Table 6 Update times of cells affected by a sensor

The update time for cells affected by sensor depends on the sensor's range and on the number of sensors being used. When a certain number of cells are affected by several sensors, the update time for these cells than longer as the same number of cells when affected by a single sensor. When too many cells are affected by sensors and the time when a sensor sends a new update to the server is very short, a communication error with the database could appear.

The time to update every cell that is affected by a certain rule depends on the area of the rule. As long as the update time for every cell is shorter than the update of the rule, there will not be a problem. When the update time of a rule is shorter than the update time of every cell which is affected by the rule, a communication error could appear and the values within a cell are not necessarily correctly calculated. It can happen that a rule uses old values of a cell. In table 7 the update times for a certain number of cells are given.

Number of cells	100	5000	50.000	150.000
Update time in milliseconds	4	276	1115	2060

Table 7 Update speed rules

In the visualization, several requests to the server are made. Each request sends the boundaries of a particular area to receive the highest value of every cell within it. The boundaries and number of areas are based on the current zoom factor, the pixelation of the area of interest, a draw factor and the size of the browser window. The higher the draw factor is, the faster the calculations can be done and the less communication is needed. But the accuracy of the visualization is reduced when the draw factor increases. Tables 8, 9 and 10 show the average time from a request to the database until the completion of drawing the areas within the visualization. As expected the time decreases when the draw factor increases. The number of cells depends on the draw factor and on the zoom factor. It is interesting that the zoom factor leads to more areas but the time to draw the areas is nearly the same. The reason for this effect is that the bottleneck is the calculation of each area and not the communication between the client and the database or the SQL command to select the highest value within a given range.

Number of areas	1	12	32	112	182
Average time in milliseconds	2080	1540	1663	1764	1771

Table 8 Request time of the visualization, draw factor equals one

Number of areas	1	12	56	70
Average time in milliseconds	456	823	763	1042

Table 9 Request time of the visualization, draw factor equals two

Number of areas	1	3	21
Average time in milliseconds	161	420	408

Table 10 Request time of the visualization, draw factor equals three

When the update time of the visualization is faster than the average draw time, the visualization maybe hard to understand, because the visualization may start to redraw the areas directly after they are finished with drawing. When an effect like this appears, the user can change the update speed within the rectangle.html file. How to do this is explained in the user guide. By default, the visualization gets updated every five seconds.

5 Conclusions

Search and rescue missions will be still important in the future. To be able to use modern technologies like UAVs optimally it is necessary to establish tools now which are capable to use the full potential of unmanned vehicles, because drones can offer a wide range of new possibilities for this special research field, such as finding shipwrecked people automatically or to localize possible threats in real time during a military operation. A decision maker could use the information provided by a tool to improve his choices. A short overview of such tools was given in section 2.5 and this work explained a tool which can use the information provided by unmanned vehicles to automatically create probability distribution maps. But this tool is not limited to data from drones. It can use all given information as long as that information, which is send to the server, has the correct format. It is also possible to predefine the probability distribution map and to create simple logical behavior for items of interest within specific regions to simulate movement of the items. How this works is explained in chapter 3. In chapter 4 it is shown that the tool can work in real-time as long as the resolution is not too precise and the area of interest is not too large.

Summing up, this tool provides an easy to understand graphical user interface which provides a decision maker in real time with important data about a search area. The decision maker can observe the movement of sensors, no matter if the sensor is a drone or controlled and moved by a real person, the chance if the target of interest is moving based on his default settings.

5.1 Limitations of this Work

It was not possible to test the tool in a real environment because the necessary hardware was not available. However, a simulation was created and added to the tool. With the simulation, it was possible to test its functionality.

As already discussed in Chapter four the tool has some issues. Depending on the results from a live test it is maybe necessary to make improvements. Some weak points in the tool could be the usability, limitations of the logical behavior and performance problem for big area of interests with a small resolution.

5.2 Future Work

Next the tool should be tested in a real scenario to see if everything is working as intended and if the tool can actually provide a decision maker with useful information. Based on these tests the tool might be improved further.

To use the tool, the files can be easily placed within the same directory of a server. It is necessary to make a connection to a database and import the required files to the database. This is explained in more detail in the user guide. When the server is running, the user can initialize the area of interest. The recommended size of the area depends on the pixelation. For firsts test the number of cells should not exceed 10.000. When a square is used it is possible to cover an area of approximate 1km^2 with a pixelation of 0,0001. This means each cell is approximate 10m times 10m large.

The initialization value can be every number between zero and one. The values for each cell gets updated. In the model, it is presumed that every cell has an adjacent cell. When a cell is on the border of the area of interest the missing adjacent cell is assumed to be a copy of the cell. This could cause interference depending on the logical rules and the real initialization values. When the tool is tested in a real-life scenario it is necessary to check, if this effect have an influence on the search mission.

The glimpse probability of a sensor depends on the accuracy of the sensor. During this work, it was not possible to make live tests with sensors so a presumption of a good value for the glimpse probability cannot be made. The update time for each sensor has no dependencies so it is possible that one sensor updates every few seconds and another sensor updates only every few minutes. Each sensor needs only to send its data to the server as explained in chapter 3.4.4.

For the logical rules for first tests it is recommended to use only a few rules in the beginning. Later on, the complexity could be increased and the rules could be more accurate but to test the functionalities it is sufficient to use only a single rule. The speed of an item of interest for each rule depends on the item and the area. The speed is measured in m/s. An object with a speed of 1 would move with 3,6 km/h and this value could be used for person who is moving by feet in a built-up area. The speed of an ocean current is usually between 0,1 m/s and 2 m/s.

When first tests are made it could appear that the Markov Chain approach to evolve the values of the cells over time may cause some troubles. A reason for this could be different update times between different rules which leads to different influence factors of each cell and interfere with more than one item of interest within the area. An idea which was made during the creation of the work is to evolve each cell with a Gompertz function. The Gompertz function is an asymmetric saturation function and is used to model biological growth. This function could be used to change the mathematical model in a way, that cells with a very low or very high probability have a smaller influence on adjacent cells.

6 Annex

In this chapter, the Annex for the thesis is presented. There is a User guide which explains how to use this tool and a list of which digital files are needed to use the tool.

6.1 User Guide

The user guide explains how to install the tool on a server. After that is an explanation on how to use the tool.

To use the tool, install the required software first. It is necessary to have NodeJS and MySQL installed on the server. All needed files and instructions to install NodeJS on the user's operation system are found here https://nodejs.org/en/download/package-manager/. It is also necessary to download and install MySQL. It is recommended to install the MySQL Workbench when the operating system is able to use a graphical user interface but it is not required. All parts of MySQL can be found here https://dev.mysql.com/downloads/.

After NodeJS and MySQL are installed, install the MySQL driver for NodeJS. A guide how he can do this is given on this website https://dev.mysql.com/downloads/. It is also necessary to be able to use ASP.NET scripts on the server. A guide on how to deploy an ASP.Net Core app to a hosting environment is given here https://docs.microsoft.com/en-us/aspnet/core/publishing/?tabs=aspnetcore1x.

When everything has been installed successfully the user must copy the folder with the scripts in a location where the server can use them. For example, when Microsoft Internet Information Services (IIS) are used to run the server it is necessary to deploy the folder within the *"wwwroot"* directory. The name of the script folder is irrelevant. In the digital annex, the folder with the scripts is called *"thesisScripts"*. In this folder, it is also required to have the folder *"js"* and *"node_modules"*. The script files *"rectangleGetData.asp"* and *"rectangleMap.html"* may be placed somewhere else as long as they are in the same directory and the .asp file has access to the MySQL database.

After the files are in the correct directories, set the correct settings for the connections between the server and the database. First, create the connection for the visualization. To do so, open the *"rectangleGetData.asp"* file. In line 21 the connection is made. The user must to change the string in this line. An explanation about database connection with asp pages is given here https://support.microsoft.com/en-us/help/300382/how-to-create-a-database-connection-from-an-asp-page-in-iis.

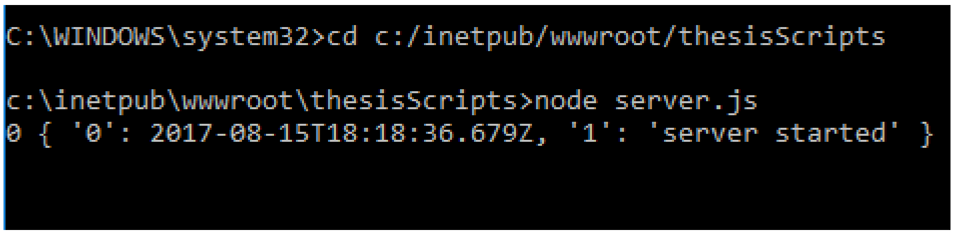
Second, the connection of the server to the database is made. To do so, open the *"server.js"* file. Between line 217 and 222 the settings for the database connection are done. Currently the password is saved directly in the code. It is possible to change that to have a higher security level. In line 218, the user needs to set the IP of the host. In the next lines, the user name and his password are set. They should be the same as in the MySQL database. In line 221 the name of the database is set. By default, the name is *"probabilitymap"*. When the user wants to use another name, it is also required to change this name within the MySQL database.

In line 215, set the port where the server is listening for user input.

When the connections between the scripts and the database are made, it is necessary to create the tables in the database. To do so, create two tables *"dronelist"* and *"mapsettings"* as described in sections 3.4.1 or import the database from the digital annex.

In order to delete an area of interest, delete the entry in the table *"mapsettings"*. It is necessary to delete every entry in the table *"dronelist"* which has a reference to the deleted area of interest. This is every row, where the type has the same name as the deleted area of interest. It is recommended to stop the server when an entry is being deleted. When it is not possible to stop the server, updates of the area of interest should be deactivated. And when it is possible that a sensor which affects this area could send new data, the reference of this sensor should be deleted first. After that, delete the tables which contains the name of the area of interest. The first table has only the name of it, the second table has the name of it and a *"rule"* suffix.

When the setup is done, the user can start the server (figure 35). While



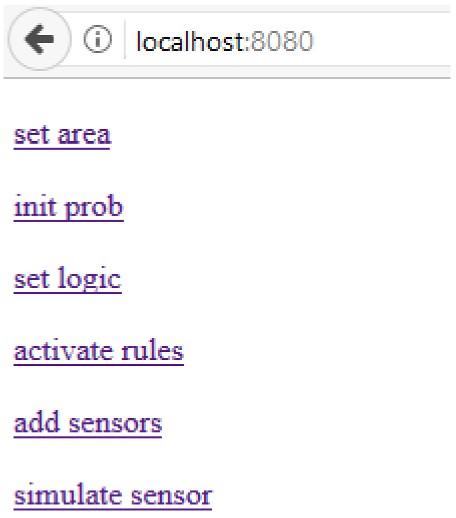
```
C:\WINDOWS\system32>cd c:/inetpub/wwwroot/thesisScripts

c:\inetpub\wwwroot\thesisScripts>node server.js
0 { '0': 2017-08-15T18:18:36.679Z, '1': 'server started' }
```

Figure 35 Start server

doing it, ensure that the server has writing access to the directory where the script is because the server will deploy a log file there. After the server is running, access the server over the IP and the port which was set in the server file.

The user is direct to the index file (figure 36). Access the config files here. In the first step, create a new area of interest. To do so, use the *"set area"* link. You will be directed to a new page (figure 37). In this page, create or load an area of interest. In order to load an existing area, type in the name of this area in the field *"source name"*


localhost:8080

set area

init prob

set logic

activate rules

add sensors

simulate sensor

Figure 36 Index file

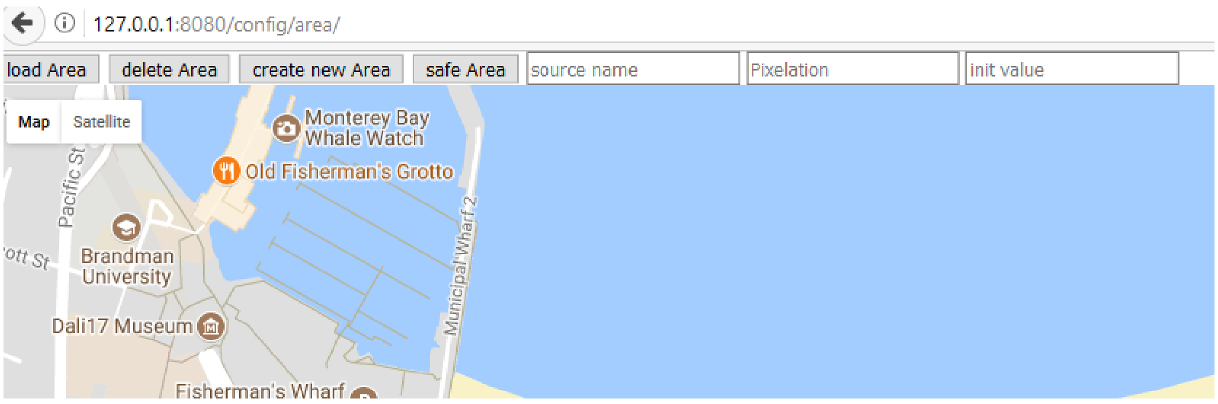
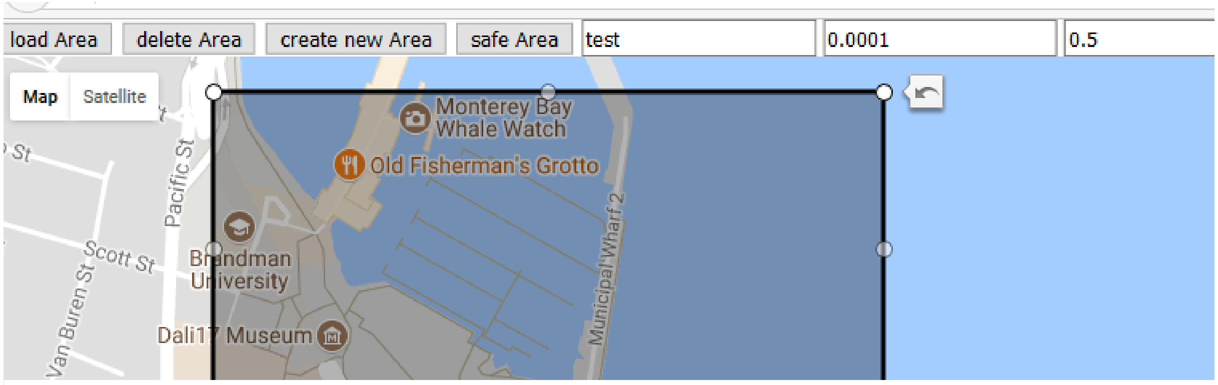


Figure 37 Create new Area

and click on the button *"load Area"*. To create a new area, click on the button *"create new Area"*.

A rectangle will appear in the middle of the screen after the user loaded an existing area or created a new one. Move this rectangle around and change its size to cover the desired area



test

0.0001

0.5

Figure 38 Safe new Area

of interest. Click on *"delete Area"* and the current area will be deleted. This will not delete the table in the database when an area was loaded. In order to save the area, click on the *"safe Area"* button. It is required to have values in all three fields (figure 38). The *"init value"* is necessary to initialize the probability map correctly and change the values later on. It is recommended to use values between zero and one.

After the area is saved, navigate back to the index file. Here you can change the initialization values for the created area of interest by clicking on *"init prob"* or create logical rules for this area with the link *"set logic"*.

To change the initialization values of the area use the respective link and a new interface will open (figure 39).

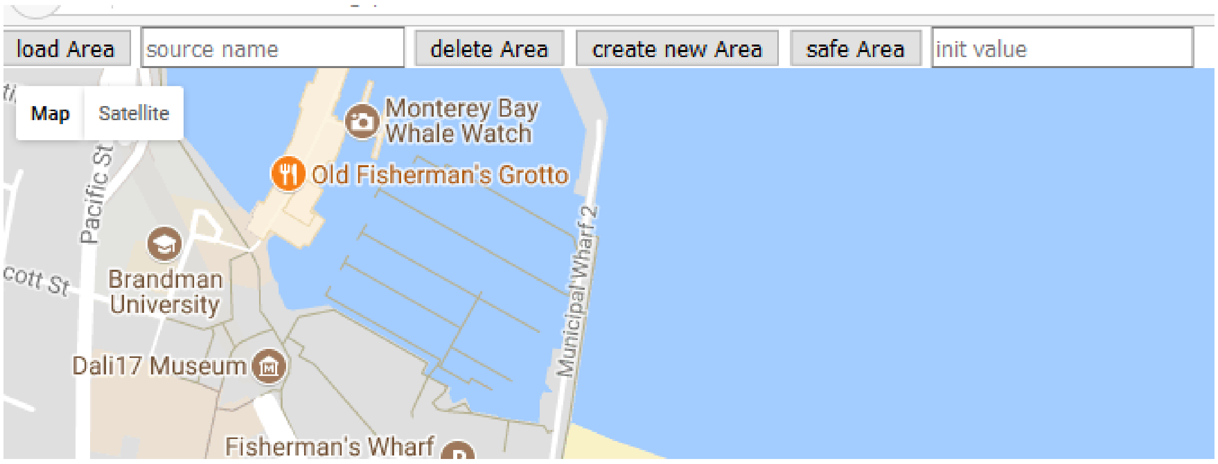


Figure 39 Initialization of Probability map 1

Load a specific area of interest within this interface by typing in the name and clicking on the button, or create a new area. This area can be dragged over the area of interest and can be used to select regions to change the initialization values (figure 40).

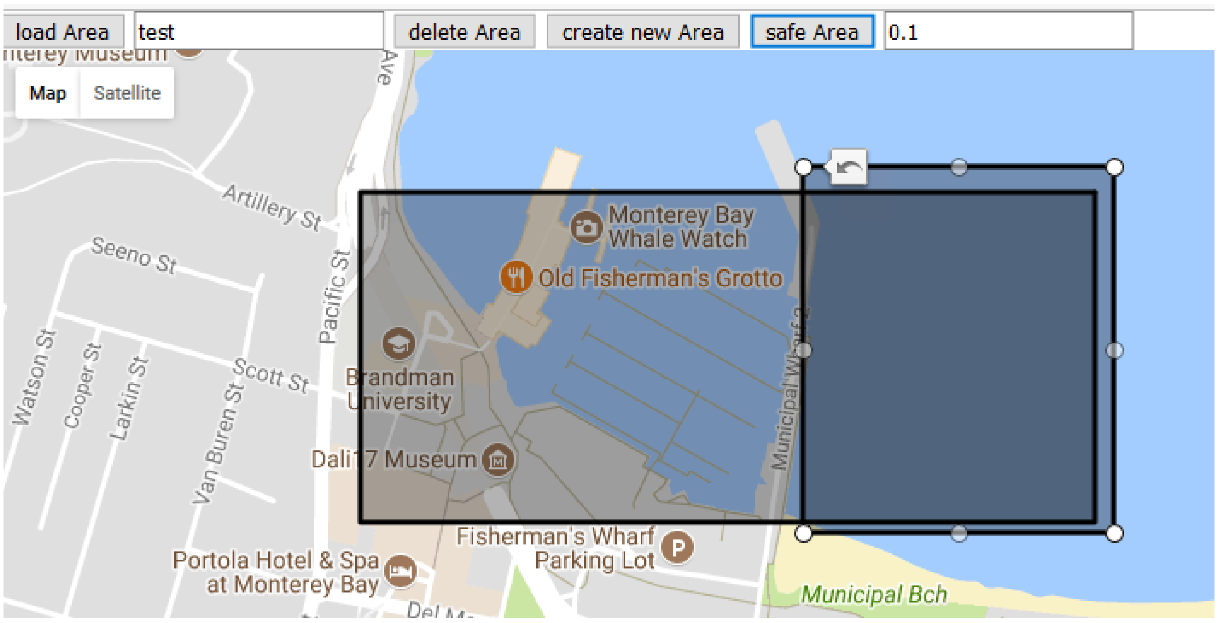


Figure 40 Initialization of Probability map 2

When the values change for a specific area, it can be understood that a sensor has observed the subarea. The probability that it has found something within this subarea is equal to the value in *"init value"*.

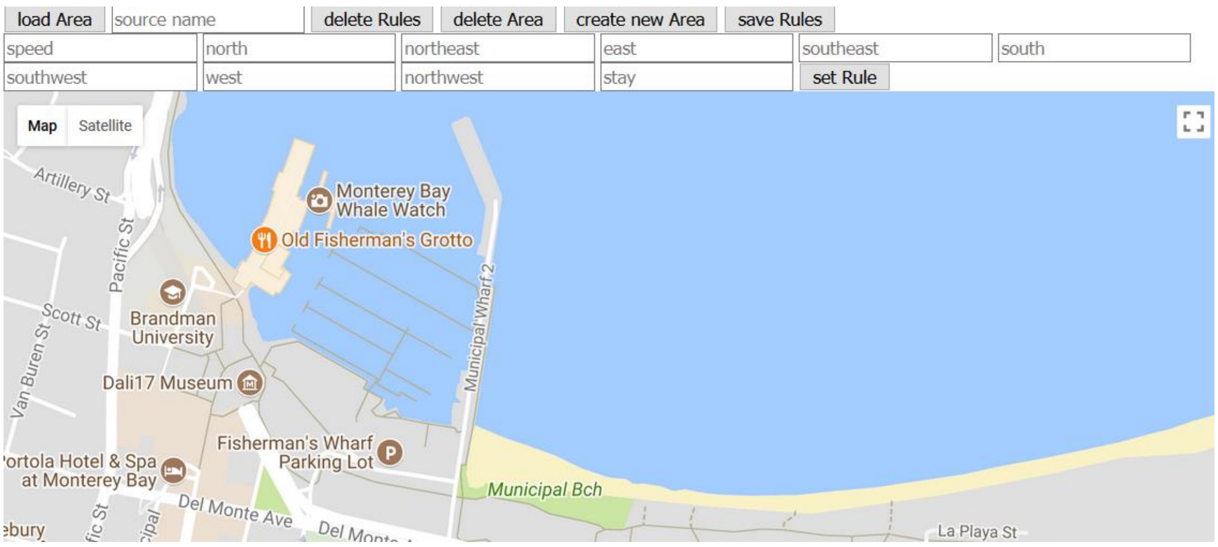
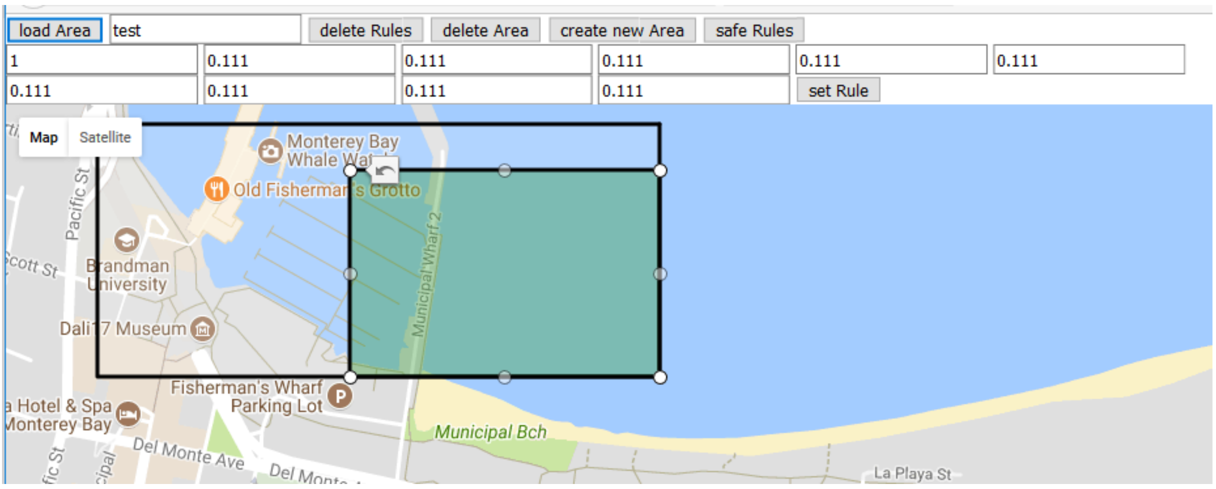


Figure 41 Interface logic rules 1

The interface for the logical rules is slightly different (figure 41). You can also load a specific area of interest. When loading an area, every existing rule is loaded, too. Values can be set for every movement direction of an item of interest within a subarea. Create as many subareas as desired. When creating new rules, it is recommended to delete all other rules first by clicking the *"delete Rules"* button. When creating a new area, drag this area around and set the wanted values for the rule within this area. Before switching to another area, click the *"set Rule"* button. Otherwise the rule will not be stored. A rule will not be saved immediately when the rule is set. To save the rules it is necessary to click on the "*save Rules*" button. Now all the rules are sent to the server and a stored in the database. A single area can also be deleted with the *"delete Area"* button.



load Area | test | delete Rules | delete Area | create new Area | safe Rules

1 | 0.111 | 0.111 | 0.111 | 0.111 | 0.111

0.111 | 0.111 | 0.111 | 0.111 | set Rule

Figure 42 Interface logic rules 2

The selected area is always displayed in a green color. Select another area by clicking on it (figure 42). The white area is the area of interest and cannot be moved or modified in this part.

Figure 43 shows the interface to activate and deactivate a specific area of interest. When an area is deactivated the rules within this area will not be applied.

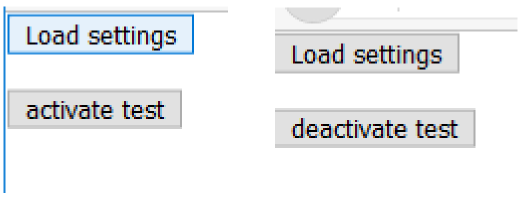
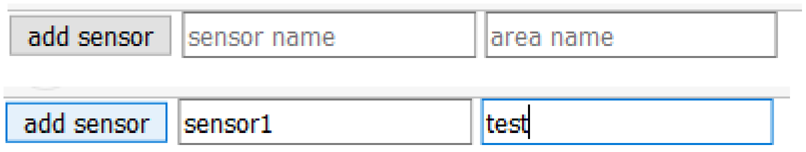


Figure 43 Activate and deactivate an area

In order ro add sensors, follow the *"add sensors"* link within the index file. Here (figure 44), set a name for a sensor and the area of interest which is affected by this sensor. When a sensor can affect more than one area, the sensor must be added for every area of interest. When a sensor is added and the database already has an entry for this sensor the old entry is deleted.



add sensor | sensor name | area name

add sensor | sensor1 | test

Figure 44 Add sensors

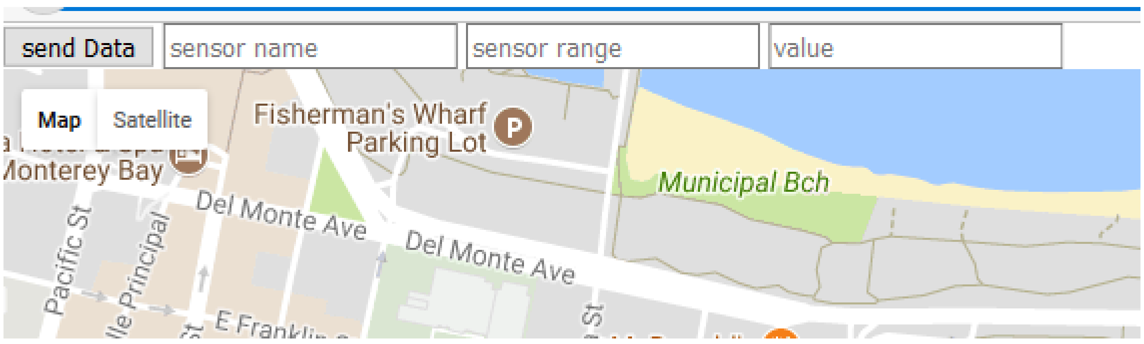


Figure 45 Sensor simulation 1

To simulate a sensor, open the interface for it (figure 45) and type in the name of the sensor the range of it in metres and the value of it. Type in the value with the glimpse probability. When a sensor detects something the value would be *"one"* and when the glimpse probability is 100% the value would remain *"one"* (figure 46).

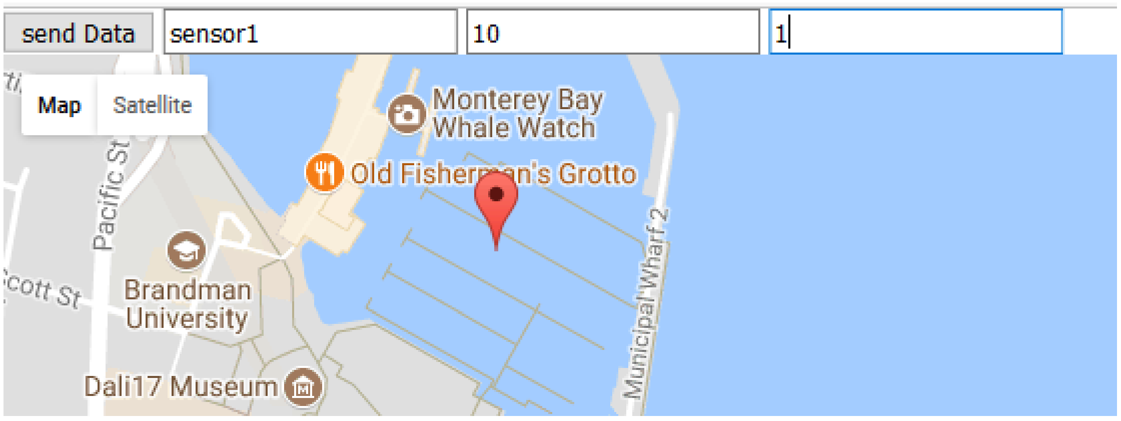


Figure 46 Sensor simulation 2

The area of interest can be observed in the visualization. It cannot be accessed over the index file; open it separately. In this file, load an area of interest and set a draw factor for this area (figure 47). The draw factor can be changed every time and the visualization will be change at the next update time.

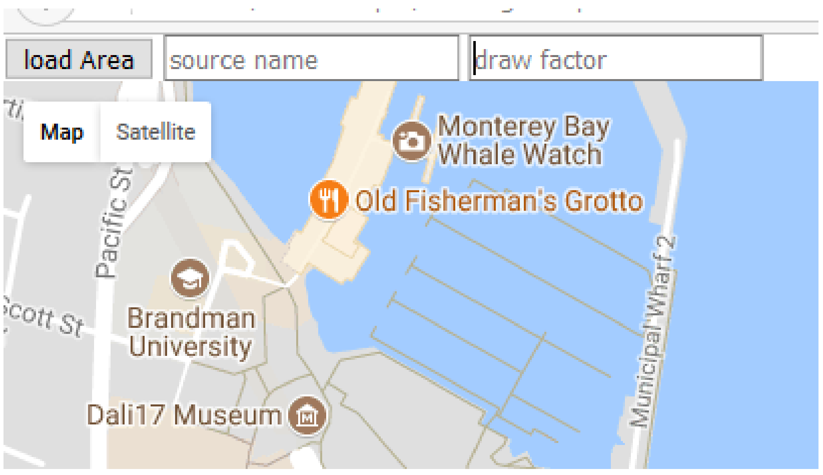


Figure 47 Observe area of interst 1

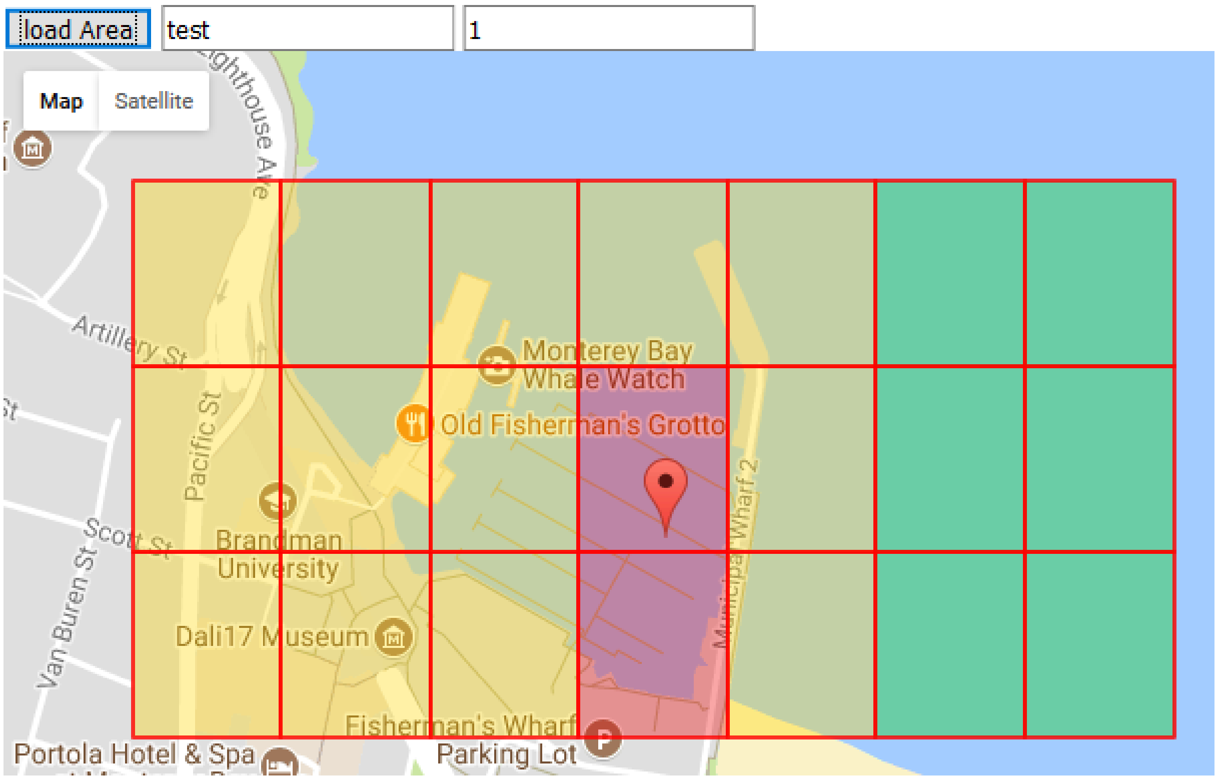


Figure 48 Observer area of interest 2

In figure 48 the observation of an area of interest is shown. This picture shows the area of interest which was created during the explanation within this user guide. The marker in the middle shows the last position of the sensor. When the sensor would have more than one position, there would be a blue line between all positions of the sensor.

6.2 Digital Files

In the digital annex, several files should be in the folder. A list of the necessary files is provided here:

Name	Purpose
js (folder)	A folder with necessary plugins for the js scripts.
Node_modules (folder)	A folder with necessary modules for NodeJS.
configArea.html	The file to create an area of interest.
configDrone.html	The file to create sensors for specific areas.
configLogic.html	The file to set specific rules for an area of interest.
configProb.html	The file to initialize the probability map.

configUpdate.html	The file to activate and deactivate the rules for specific areas of interest.
database.sql	The scheme for the MySQL database. This scheme can be imported to the user’s database.
debug.txt	A debug file. The server will store its logs into this file.
Index.html	The index file to navigate through the config menu.
package.json	A file for the communication between the clients and the server.
rectangleGetData.asp	The ASP.NET file the visualization needs to access the database.
rectangleMap.html	The visualization to observe an area of interest.
sensorSim.html	The file to simulate sensor input.
server.js	The server

7 Bibliography

[1] B. R. Rodriguez (2016), *3800 Tote im Mittelmeer – allein in diesem Jahr*, Spiegel Online, http://www.spiegel.de/politik/ausland/fluechtlinge-im-mittelmeer-sind-2016-mehr-menschen-gestorben-als-je-zuvor-uno-a-1118441.html, Last accessed: 04.08.2017

[2] (2016), *Zahl der ertrunkenen Migranten auf Rekordstand,* Spiegel Online, http://www.spiegel.de/panorama/fluechtlinge-im-mittelmeer-zahl-der-ertrunkenen-migranten-auf-rekordstand-a-1127373.html, Last accessed: 04.08.2017

[3] J. Lutteroth (2014), *Die Jahrtausend-Katastrophe,* Spiegel Online, http://www.spiegel.de/panorama/gesellschaft/tsunami-2004-in-suedost-asien-die-grosse-flut-a-1006392.html, Last accessed: 04.08.2017

[4] J. Reschke, *Erdbeben in Haiti,* http://www.das-erdbeben.de/haiti.htm Last accessed: 04.08.2017

[5] T. Kratzke, L. Stone, J. Frost (2010), *Search and rescue optimal planning system*, In Proceedings of the 13th Conference on Information Fusion

[6] A. Ohsumi (1984), *Stochastic control with searching a randomly moving target*, In Porceedings of the 1984 American Control Conference

[7] N. Yau (2011), *Visualize This: The FlowingData Guide to Design, Visualization, and Statistics,* Jon Wiley & Sons

[8] P. G. W. Keen (1980), *Decision support systems: a research perspective*, Cambrige, Mass.: Center for Information Systems Research

[9] D. Schuff, D. Paradice, F. Burstein, D. J. Power, R. Sharde (2010), *Decision Support an Examination of the DSS Discipline*, Springer Science & Business Media

[10] P. Haettenschwiler (1999), *Neues anwenderfreundliches Konzept der Entscheidungsunterstützung. Gutes Entscheiden in Wirtschaft, Politik und Gesellschaft,* Zürich vdf Hochschulverlag AG

[11] D. J. Power (2002). *Decision support systems: concepts and resources for managers.* Greenwood Publishing Group

[12] P. K. Davis, J. Kulick, M. Egner (2005), *Implications of Modern Decision Science for Military Decision-Support Systems,* Rand Corporation http://www.rand.org/content/dam/rand/pubs/monographs/2005/RAND_MG360.pdf, Last accessed: 04.08.2017

[13] E. Susnea (2012), *Decision Support Systems in Military Actions: necessity, Possibilities and Constraints*, Journal of Defense Resources Management Vol. 3 http://journal.dresmara.ro/issues/volume3_issue2/12_susnea.pdf, Last accessed: 04.08.2017

[14] T. Ritchey (2014), *Four Models about Decision Support Modeling,* Acta Morphologica Generalis, http://www.amg.swemorph.com/pdf/amg-3-1-2014.pdf, Last accessed: 04.08.2017

[15] A. Waal, T. Ritchey (2007), *Combining morphological analysis and Bayesian networks for strategic decision support*, Swedish Morphological Society, http://www.swemorph.com/pdf/mabn.pdf, Last accessed: 04.08.2017

[16] D. Doty, W. Glick (1994), *Typologies as a unique form of theory building,* Academy of Management

[17] C. Nyce (2007), *Predictive Analytics White Paper,* American Institute for CPCU/ Insurance Institute of America, http://www.hedgechatter.com/wp-content/uploads/2014/09/predictivemodelingwhitepaper.pdf, Last accessed: 04.08.2017

[18] L. Moreira-Matias, J. Gamba, M. Ferreira, J. Mendes-Moreira, L. Damas (2016), *Time-evolving O-D matrix estimation using high-speed GPS data streams,* Expert Systems with Applications vol. 44

[19] W. Eckerson (2007), *Predictive Analytics,* Transforming Data With Intelligence, https://tdwi.org/articles/2007/05/10/predictive-analytics.aspx?sc_lang=en, Last accessed: 04.08.2017

[20] E. Siegel (2016), *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die,* John Wiley & Sons

[21] S. Korn (2011), *The Opportunity for Predictive Analytics in Finance,* HPC wire, https://www.hpcwire.com/2011/04/21/the_opportunity_for_predictive_analytics_in_finance/, Last accessed: 04.08.2017

[22] C. D. Kirkpatrick, J. R. Dahlquist (2006), *Technical Analysis: The Complete Resource for Financial Market Technicians,* Financial Times Press

[23] K. Pearson (1905), *The Problem of the Random Walk*, Nature Band 72

[24] P. Wegmann (2005), *Überblick Finanzmarkttheorie*. Vorlesungsunterlagen, Universität Basel

[25] H. Fletcher (2011), *The 7 Best Uses for Predictive Analytics in Multichannel Marketing,* Target Marketing, http://www.targetmarketingmag.com/article/7-best-uses-predictive-analytics-modeling-multichannel-marketing/, Last accessed: 04.08.2017

[26] E. Stevenson (2011), *Tech Beat: Can you pronounce health care predictive analytics?*, Times-standard, http://www.times-standard.com/general-news/20111216/tech-beat-can-you-pronounce-health-care-predictive-analytics, Last accessed: 04.08.2017

[27] E. Felipe-Barkin (2011), *CRM + Predictive Analytics: Why It All Adds Up,* CRM magazine, http://www.destinationcrm.com/Articles/Editorial/Magazine-Features/CRM---Predictive-Analytics-Why-It-All-Adds-Up-74700.aspx, Last accessed: 04.08.2017

[28] S. Geissner (1993), *Predictive Inference: An Introduction*, CRC Press

[20] M. Rouse (2012), *descriptive modeling*, TechTarget, http://whatis.techtarget.com/definition/descriptive-modeling, Last accessed: 04.08.2017

[30] B. v. Halle, L. Goldberg (2009), *The Decision Model: A Business Logic Framework Linking Business and Technology*, CRC Press

[31] (2012), *Regression analysis,* Springer-Verlag Encyclopedia of Mathematics, https://www.encyclopediaofmath.org/index.php/Regression_analysis, Last accessed: 04.08.2017

[32] W. Härdle (1990), *Applied Nonparametric Regression*, Cambridge University Press

[33] D. A. Freedman (2009), *Statistical Models: Theory and Practice*, Cambridge University Press

[34] W. L. Hosh, *Machine learning Artifical Intelligence,* Encyclopaedia Britannica, https://www.britannica.com/technology/machine-learning, Last accessed: 04.08.2017

[35] A. Munoz, *Machine Learning and Optimization,* New York University https://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf, Last accessed: 04.08.2017

[36] K. Gurney (2003), *An Introduction to Neural Networks,* CRC Press

[37] B. T. Hazen, C. A. Boone, J. D. Ezell, L. A. Jones-Farmer (2014), *Data quality for data science, predicte analytics, and big data in supply chain management: An introduction to the problem and suggestions for research and applications,* International Journal of Production Economics vol. 154

[38] S. Finlay (2014), *Predictive Analytics, Data Mining and Big Data. Myths, Misconceptions and Methods*, Springer-Verlag

[39] D. Temple-Raston (2012), *Predicting The Future: Fantasy Or A Good Algorithm?* http://www.npr.org/2012/10/08/162397787/predicting-the-future-fantasy-or-a-good-algorithm, Last accessed: 04.08.2017

[40] G. Shmueli (2010), *To Explain or to Predict?*, Statistical Science vol. 25, Institute of Mathematical Statistics

[41] P. Toth, D. Vigo (2002), *The vehicle routing problem,* Society for Industrial and Applied Mathematics

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein (2001), *Introduction To Algorithms,* 2. Auflage, MIT Press

[43] E. W. Dijkstra (1959), *A Note on Two Problems in Connexion with Graphs,* http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf, Last accessed: 04.08.2017

[44] J. Lerner, D. Wagner, K. Zweig (2009), *Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation,* Springer Science & Business Media

[45] N. Christofides, *Worst-case analysis of a new heuristic for the travelling salesman problem*, Report 388, Graduate School of Industrial Administration, Carnegie Mellon University (CMU), 1976

[46] T. Bäck (1996), *Evolutionary Algorithms in Theory and Practice: Genetic Algorithms, Evolution Strategies,*Oxford University Press

[47] C. Blum, D. Merkle (2008), *Swarm Intelligence: Introduction and Applications*, Springer Science & Business Media

[48] H. N. Psaraftis, M. Wen, C. A. Kontovas (2015), *Dynamic vehicle routing problems: Three decades and counting,* Networks International Journal vol. 67

[49] F. Bourgault, T. Furukawa, H. F. Durrant-Whyte (2003), *Coordinated Decentralized Search for a Lost Target in a Bayesian World,* Intl. Conference on Intelligent Robots and Systems

[50] T. Ishida, R. E. Korf, *Moving Target Search,* http://www.ijcai.org/Proceedings/91-1/Papers/033.pdf, Last accessed: 04.08.2017

[51] R. E. Korf (1990), *Real-Time Heuristic Search*, Artificial Intelligence, Vol. 42

[52] M. Raap, S. Meyer-Nieberg, S. Pickl, M. Zsifkovits (2017), *Aerial vehicle search-path optimization: A novel method for emergency operations*, Journal of Optimization, Theory and Applications

[53] M. Raap, M. Zsifkovits, S. Pickl (2017), *Trajectory optimization under kinematical constraints for moving target search*, Computers & Operations Research

[54] A. Tolver (2016), *An Introduction to Markov Chains*, Department of Mathematical Sciences, University of Copenhagen http://www.math.ku.dk/noter/filer/stoknoter.pdf, Last accessed: 04.08.2017

[55] S. Stefanov (2010), *JavaScript Patterns: Build Better Applications with Coding and Design Patterns,* O'Reilly Media, Inc.

[56] C. Wenz (2007), *JavaScript und AJAX, das umfassende Handbuch,* Rheinwerk Computing http://openbook.rheinwerk-verlag.de/javascript_ajax/01_einleitung_001.htm#aa048df86f6ada81276b0bd8025c78fa, Last accessed: 04.08.2017

[57] (2016), *About Node.js, and why you should add Node.js to your skill set?,* training.com http://blog.training.com/2016/09/about-nodejs-and-why-you-should-add.html, Last accessed: 04.08.2017

[58] Node.js Foundation, https://nodejs.org/en/foundation/, Last accessed: 04.08.2017

[59] *Node.js Introduction*, https://www.w3schools.com/nodejs/nodejs_intro.asp, Last accessed: 04.08.2017

[60] Dan Kegel (2011), *The C10K Problem*, WebCite http://www.webcitation.org/6ICibHuyd?url=http://www.kegel.com/c10k.html, Last accessed: 04.08.2017

[61] *ASP and ASP.NET Tutorials*, https://www.w3schools.com/asp/, Last accessed: 04.08.2017

[62] *ASP Tutorial*, https://www.w3schools.com/asp/asp_introduction.asp, Last accessed: 04.08.2017

[63] *ASP.NET Web Pages – Tutorial*, https://www.w3schools.com/asp/webpages_intro.asp, Last accessed: 04.08.2017

[64] (2017), *DB-Engines Ranking*, solid IT GmbH, https://db-engines.com/de/ranking, Last accessed: 04.08.2017

[65] S. Pröll, E. Zangerle, W. Gassler (2015), *MySQL: Das umfassende Handbuch,* Rheinwerk Verlag GmbH

[66] (2012), *RDBMS,* Datenbanken Online Lexikon, http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/RDBMS, Last accessed: 04.08.2017

[67] J. Smith (2007), *Composite Primary Keys,* Jeff's SQL Server Blog, http://weblogs.sqlteam.com/jeffs/archive/2007/08/23/composite_primary_keys.aspx, Last accessed: 04.08.2017

[68] T. Kratzke, L. Stone, and J. Frost (2010), *Search and rescue optimal planning system*, In Proceedings of the 13th Conference on Information Fusion

[69] H. R. Richardson, D. H. Wagner, J. H. Discenza (2006), *The United States Coast Guard Computer-Assisted Search Planning System (CASP)*,* Naval Research Logistic Quarterly Vol. 27, Issue 4

[70] (2013), *U.S. Coast Guard Addendum,* U.S. Department of Homeland Security

[71] International Civil Aviation Organization and International Maritime Organization: IAMSAR MANUAL, Volume I: ORGANIZATION AND MANAGEMENT, 2007 Consolidated Edition

[72] International Civil Aviation Organization and International Maritime Organization: IAMSAR MANUAL, Volume II: ORGANIZATION AND MANAGEMENT, 2007 Consolidated Edition

[73] International Civil Aviation Organization and International Maritime Organization: IAMSAR MANUAL, Volume III: ORGANIZATION AND MANAGEMENT, 2007 Consolidated Edition

[74] M. Kress (2008), *Aerial Search Optimization Model (ASOM) for UAVs in Special Operations,* Naval Postgraduate School

[75] Ordnance Survey (2016), *A Guide to coordinate systems in Great Britannia*, https://www.ordnancesurvey.co.uk/docs/support/guide-coordinate-systems-great-britain.pdf, Last accessed: 11.07.2017

[76] (2012), *World Geodetic System 1984*, United Nations Office for Outer Space Affair, http://www.unoosa.org/pdf/icg/2012/template/WGS_84.pdf, Last accessed: 11.07.2017

[77] W. Korth (2009), *Geodynamik & Erdmessung*, http://public.beuth-hochschule.de/~korth/vorl_Erdmessung.pdf, Last accessed: 11.07.2017

[78] G. Hofbauer (2005), Alfred Wegener – *Driftende Kontinente und unbewegliche Geologen,* http://www.gdgh.de/berichte/b09/wegener.pdf, Last accessed: 11.07.2017

[79] T. Bröcker (2004), *Lineare Algebra und Analytische Geometrie: Ein Lehrbuch Für Physiker Und Mathematiker*, Springer-Verlag

[80] W. S. Peters (1985), *Satz des Pythagoras: ein abbildungsgeometrischer Beweis*, Institut für Film und Bild in Wissenschaft und Unterricht

[81] (2009), *2-Sphäre,* Wikipedia https://de.wikipedia.org/wiki/Sph%C3%A4re_(Mathematik)#/media/File:Sphere_wireframe_10deg_6r.svg, Last accessed: 04.08.2017

[82] G. V. Brummelen (2013), *Haevenly mathematics: The Forgotten Art of Spherical Trigonometry,* Princton University Press

[83] W. Torge (2003), *Geodäsie*, 2. Auflage. De Gruyter

[84] W. W. Sheppard, C. C. Soule (1922), *Practical Navigation*, World technical Institute

[85] G. A. Korn (2000), *Mathematical handbook for Scientists and Engineers*, Dover Publications Inc.

[86] (2008), *File:Law-of-haversines.svg,* Wikipedia, https://en.wikipedia.org/wiki/File:Law-of-haversines.svg

[87] A. Tolver (2016), *An Introduction to Markov Chains*, University of Copenhagen, http://www.math.ku.dk/noter/filer/stoknoter.pdf, Last accessed: 04.08.2017 markov introduction

[88] M. Kompf (2016), *Distance calculation*, https://www.mkompf.com/gps/distcalc.html, Last accessed: 11.07.2017

[89] (2011), https://gis.stackexchange.com/questions/7430/what-ratio-scales-do-google-maps-zoom-levels-correspond-to, Last accessed: 04.08.2017

[90] A. v. Kesteren, J. Auborg, J. Song, H. R. M. Steen (2016), *XMLHttpRequest Level 1,* W3C Working Group, https://www.w3.org/TR/2016/NOTE-XMLHttpRequest-20161006/, Last accessed: 04.08.2017

[91] *Google Maps for every platform,* Google Developers, https://developers.google.com/maps/, Last accessed: 04.08.2017

[92] S. W. Smith (1997), *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Pub.

[93] (2012), *How accurate is approximating the Earth as a sphere?,* Geographic Information System ,https://gis.stackexchange.com/questions/25494/how-accurate-is-approximating-the-earth-as-a-sphere#25580, Last accessed: 04.08.2017

[94] JavaScript Numbers, https://www.w3schools.com/js/js_numbers.asp, Last accessed: 04.08.2017

[95] *B.5.2.10 Packet Too Large,* Oracle Corporation, https://dev.mysql.com/doc/refman/5.7/en/packet-too-large.html, Last accessed: 04.08.2017

Erklärung

Gemäß Beschluss des Prüfungsausschusses für die Fachhochschulstudiengänge der UniBwM.

Hiermit versichere ich, dass die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden.

Ferner habe ich vom Merkblatt über die Verwendung von Masterabschlussarbeiten Kenntnis genommen und räume das einfache Nutzungsrecht an meiner Masterarbeit der Universität der Bundeswehr München ein.

Datum, Unterschrift