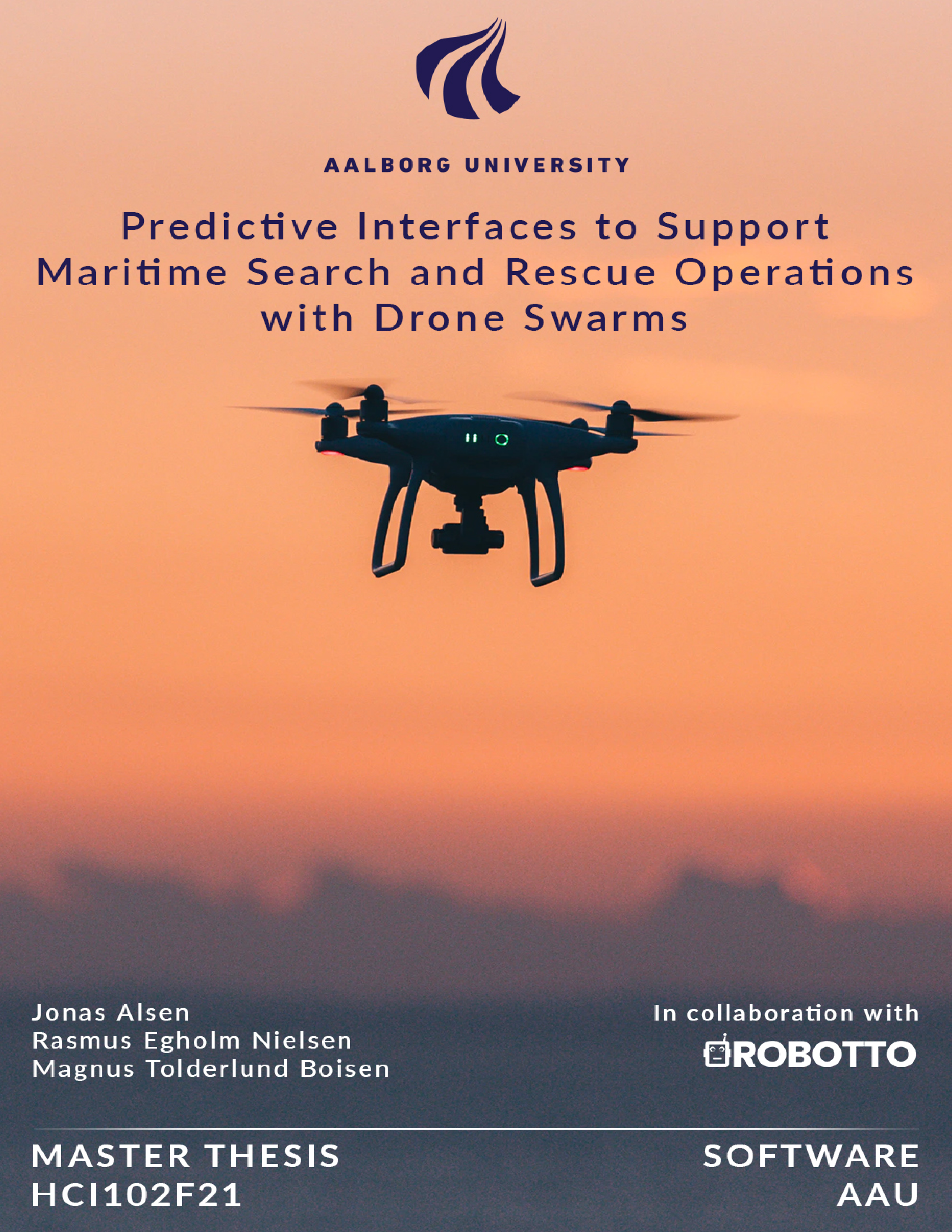


AALBORG UNIVERSITY

Predictive Interfaces to Support Maritime Search and Rescue Operations with Drone Swarms

Jonas Alsen
Rasmus Egholm Nielsen
Magnus Tolderlund Boisen

In collaboration with
ROBOTTO

MASTER THESIS
HCI102F21

SOFTWARE
AAU

AALBORG UNIVERSITY
STUDENT REPORT

Title: Predictive Interfaces to Support Maritime Search and Rescue Operations with Drone Swarms

Theme:
HCI

Project Period:
Spring Semester 2021

Project Group:
HCI102F21

Participants:
Jonas Grann Alsen
Magnus Tolderlund Boisen
Rasmus Egholm Nielsen

Supervisor:
Timothy Robert Merritt

Copies: 8
Page Numbers: 135
Date of Completion:
June 8, 2021

Synopsis:

This project is a continuation of a 9th semester project, where a multi-UAV interface was created to assist in maritime search and rescue efforts. As the current search process of people in waterways is slow and expensive, an alternative process is presented where the search helicopter is replaced by a swarm of UAVs, reducing the search time and cost.

Existing interfaces and research in the field are analyzed, and their findings suggest that predictive elements can positively influence workload and situational awareness. This field is still in its infancy and therefore could benefit from research that attempts to develop interfaces to support emergency personnel and learn from the use of these interfaces to inform the design of more effective SAR technology.

Our work focuses on the design of the interface including the information elements as well as predictive elements that might support the Operations Manager in their role. A demo system is implemented as well to simulate different rescue scenarios during the study.

User studies were conducted on several experts in the field to evaluate the interface and the predictive elements. Users preferred having the predictive elements to assist them in drawing the search area, but the results suggest that none of the predictive elements were the best and could be further improved. Overall the experts were positive about the simulated scenarios and the interface. Feedback from the experts gave future work that should be considered in such interfaces.

The findings concludes that further research should be conducted on predictive elements for SAR interfaces.

Contents

Preface

This report presents our master thesis project by HCI102F21 from Aalborg University. We all have a background in developing interfaces and are familiar with UAVs. The project was inspired by living in Aalborg, where we noticed helicopters searching along Aalborg Harbor and wanted to research further to help improve the effectiveness of search and rescue to save lives, while at the same time, look for ways technology might reduce the cost of operations.

We want to thank our supervisor Timothy Robert Merritt for his guidance and feedback throughout the semester. Additionally, we thank Robotto for their collaboration with the project and help shape the project.

Lastly, we want to thank all of our participants who took their time to participate in our user study and gave feedback to our system from their expertise.

Aalborg University, June 8, 2021

[illegible]	Magnus Boisen	[illegible]
Jonas Grann Alsen	Magnus Tolderlund Boisen	Rasmus Egholm Nielsen

1 | Introduction

This chapter will describe the motivation for the project, followed by a summary of the previous project that this work builds upon. Finally, the goals of the project will be presented.

The research of this project is based on building a suitable interface that provides a system to further investigate the performance factors in a UAV interface system for an operation manager. Through expert reviews and research on state of the art from (hci901e20 et al., 2021) a conventional interface is proposed.

1.1 Motivation

UAVs are increasingly in demand and being used in several industries like agriculture, emergency services, energy industries, and more. From 2021 to 2028 it is expected that the UAV market share will expand with 57.5%. With the increase in demand, the research on HCI related to controlling and monitoring the UAVs is becoming more relevant (Research, 2021).

One of the companies using UAVs as part of their solution is Robotto, which is a company based on a bachelor project from Aalborg University (Robotto, 2020). They developed a UAV system for detecting wildfires, accompanied by an interface that can control the UAVs in the field. They have received multiple funding and rewards for their solution, and they are experienced within the field. This collaboration gathered inspiration and design guidelines through their knowledge. Robotto has shown an interest in expanding their system to assist in maritime search and rescue (SAR) operations.

Every year people go missing near the water, and a SAR operation is urgently needed to save a drowning person's life. In 2019 in Denmark, 431 SAR operations were carried out, and 16 people lost their lives, according to (Forsvarsudvalget, 2019). Time is critical in a SAR operation with a person drowning where a few minutes can be the difference between life and death.

SAR operations today are carried out by the local emergency services with the help of the Danish military when needed. The local emergency services can monitor the whole operation and send out a rescue boat. Meanwhile, the Danish military can send a helicopter, aiding in finding the missing person. An illustration of this process can be seen in Figure 1.1. According to (Regionshospital Nordjylland, 2015) it takes around 8-12 minutes to launch a manned aircraft in the Danish military, which is a critical time where a system of UAVs can be deployed instantaneously and aid the SAR operation. Another aspect of having a UAV system is the cost of a SAR operation using a helicopter. According to (Ørsborg Johansen, 2020), a SAR operation costs the Danish military approximately 60.000 DKK per hour. This cost is per flight, whereas a UAV system will have a steep setup cost but will be cheaper over time.

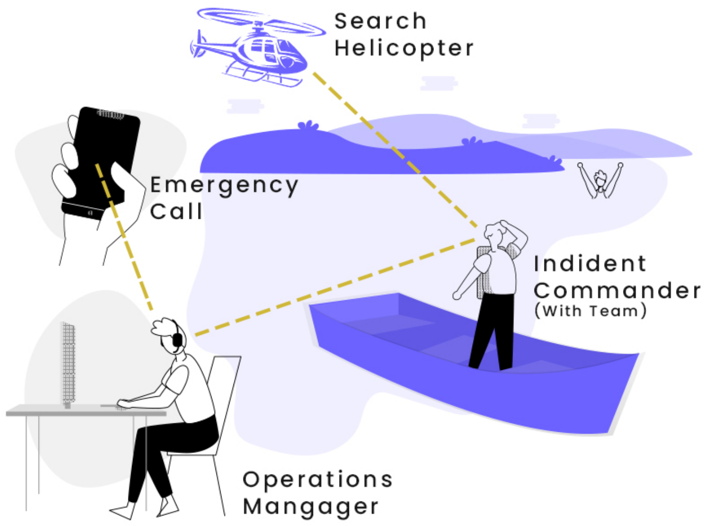


Figure 1.1: The existing rescue process

A system of multiple UAVs along the side of the shore that are sent out instantaneous scouting the area as the emergency call is received, spending the critical time locating the missing person until the helicopter arrives to support the rescue. An illustration of this new approach can be seen in Figure 1.2. The system provides an interface for controlling and monitoring the operation where a small misjudgment by an operator, caused by human factors, will result in wrong decisions and have fatal consequences. This is why a predictive interface is needed for balancing the human factors for the operator to prevent human errors.

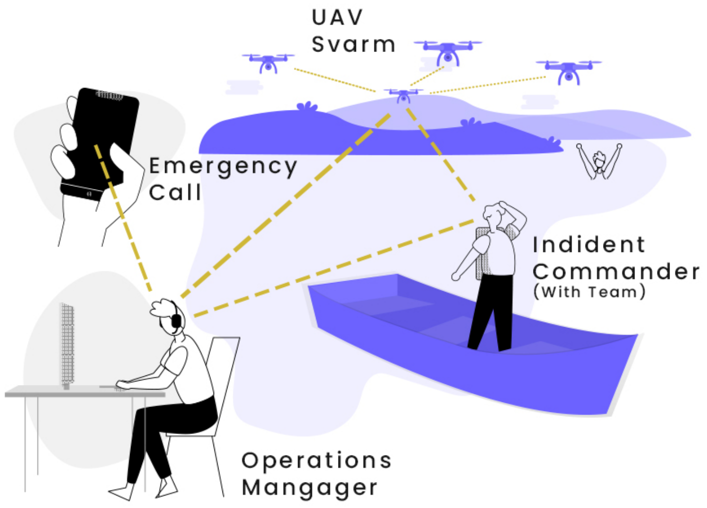


Figure 1.2: The UAV rescue process

1.2 Previous Work

This report is a continuation of a previous 9th semester project (hci901e20 et al., 2021). The project focused on developing a system that could aid rescue personnel in locating missing persons in maritime environments by utilizing a swarm of autonomous UAVs. Two distinct system operators were found through two interviews with the emergency services personnel, which were responsible for conducting rescue operations. These interviews acted as the basis for creating a set of requirements for a base system that could be used by both operators and could be extended in the future. This base system was then used in two expert reviews to gather feedback and suggestions for further improvements, and in the end, creating a new set of requirements tailored for each of the two operators.

Interviews with Operators

Two interviews were conducted to understand who would be using the system during a rescue procedure: one with an Incident Commander and another with an Operations Manager, both employed at Nordjyllands Beredskab. These interviews presented some information about each of the operators' responsibilities during a rescue operation and outlined the rescue process from start to finish. The workflow of a typical rescue operation can be seen in Description 1.

1. **Emergency Call** A call to the Danish emergency hotline about a person missing or in distress is received. The information is then forwarded to the Danish emergency services.
2. **PICK technique** The emergency hotline receives the information about the mission and uses their PICK method, which quickly and effectively starts the correct operation.
3. **Emergency Services** If the operation requires a search and rescue operation, the Danish emergency services are contacted. The Operations Manager receives the call and information on the computer simultaneously and contains necessary information like the GPS location of the incident, a description of the person missing, and important timestamps that the person was last seen.
4. **Automatic Route Planning** A route is automatically planned and alerted to the on-duty team.
5. **The Incident Commander is contacted by the Operations Manager** Operation Manager informs the Incident Commander of the relevant information about the operation.

Description 1: Workflow description of each step of a Search and Rescue operation (hci901e20 et al., 2021).

Developing a Base System

A base system was developed that both operators could initially use and could be expanded further in the future. The requirements for the base system were gathered from the interviews. The base system consists of an ASP.NET backend, an Angular frontend, and an embedded system using Python. The backend can communicate with both the embedded system and the frontend, which allows the frontend to receive data from the UAVs on the embedded system and send specific commands to each UAV.

Expert Reviews of Base System

The developed base system was evaluated in two expert reviews to validate the requirements and detect potential issues. Two experts were chosen to review the system; the first was Anders la Cour-Harbo, who is an associate professor at AAU in the Department of Electronic Systems and also the manager for the AAU Drone Research Laboratory. The second expert chosen was Jacobo Domingo Gil, the COO and Co-Founder of Robotto, a UAV specialized company used to detect wildfires. The expert reviews provided excellent inside knowledge from users within the field of work, and they uncovered problems in the current design and discovered opportunities not yet implemented.

From the expert reviews, several additional requirements were discovered for each of the operators. The missing requirements for the Operations Manager will be implemented this semester. The design and implementation of those requirements can be seen in Section 4.2.

1.3 Project Goals

This project aims to contribute to the research on human swarm interfaces. A research gap has been found through research and work throughout last semester and the master thesis, which is the studies on a predictive human swarm interface. The goal is to develop tools for the emergency services and explore if there are advantages in using a predictive human swarm interface in maritime search and rescue operations. The rest of the thesis is organized as follows:

Chapter 2 describes other work that is related to this project. Chapter 3 analyses the related work and defines the research problem. Chapter 4 presents the previous project and describes the design and implementation of the missing requirements. Chapter 5 describes the design and implementation of the predictive elements, the demo system, and additional features needed for the Operations Manager. Chapter 6 describes the user studies conducted. Chapter 7 presents the findings of the user study. Chapter 8 describes the evaluation of the user study and lastly Chapter 9 provides a conclusion to the thesis.

2 | Related Work

The work that informs our project can be categorized broadly into existing SAR interfaces, studies on predictive interfaces, and measuring and understanding human factors in user studies. It is essential to gather information on existing interfaces and understand the techniques used in the real world today.

2.1 Search and Rescue Interfaces

This section discusses three existing interfaces that include predictive elements and interface elements expected in a SAR interface. The two interfaces SAROPS and SARIS, are used in real world scenarios and include various tools, whereas the DroneResponse is a prototype that has been built in collaboration with firefighters.

One of the most comprehensive tools used in a SAR operation for planning is the Search and Rescue Optimal Planning System (SAROPS). It is used by the United States Coast Guard for planning and execution and provides a graphical interface as seen in Figure 2.1. SAROPS can simulate the environment during an operation and provide valuable data points, such as the likelihood of the search object being within the boundaries and detecting the object. The system can simulate the last known position of the object, and from a multitude of variables such as the wind and water currents, the simulator will display the results as a probability density map, also seen in Figure 2.1. Additionally, the simulator is also able to provide an optimal search plan from the portability map and will keep providing plans taking the previous searches into account until the object has been found or the operation is stopped (of Homeland Security, 2013).

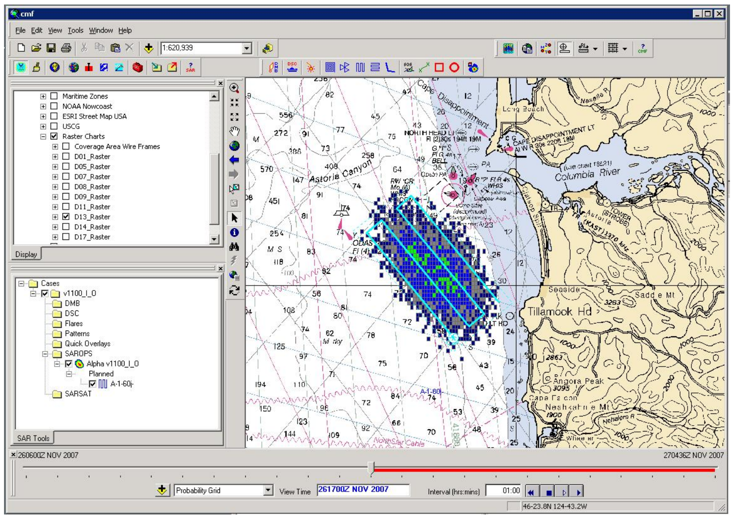


Figure 2.1: SAROPS Interface

In (Agrawal et al., 2020), a multi-UAV system was designed to help emergency responders. The system was designed in close cooperation with a fire department in a participatory design process. The system is called DroneResponse, and the interface can be seen in Figure 2.2. The design was created from the idea that situational awareness demons, which are common design errors that inhibit situation awareness, should be eliminated or minimized. These situational awareness demons were mainly used by the design team and presented at the cooperation sessions to let the firefighters give their feedback on the different demons and where they might occur. The system uses UAVs to search for missing persons, with an object detection model to notify the operator when a person is found in the water as seen in Figure 2.2.

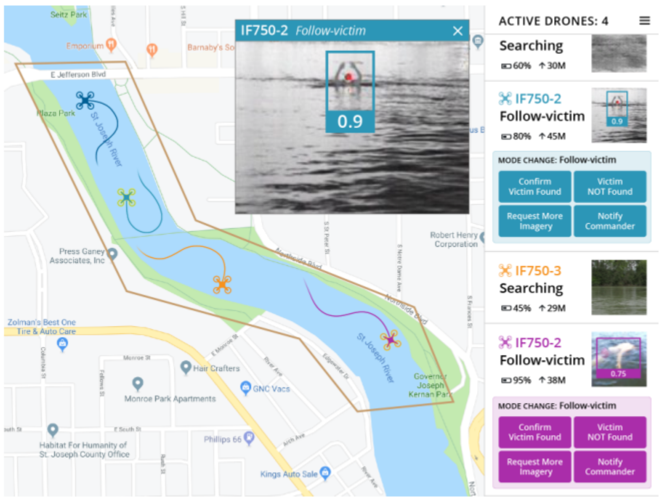
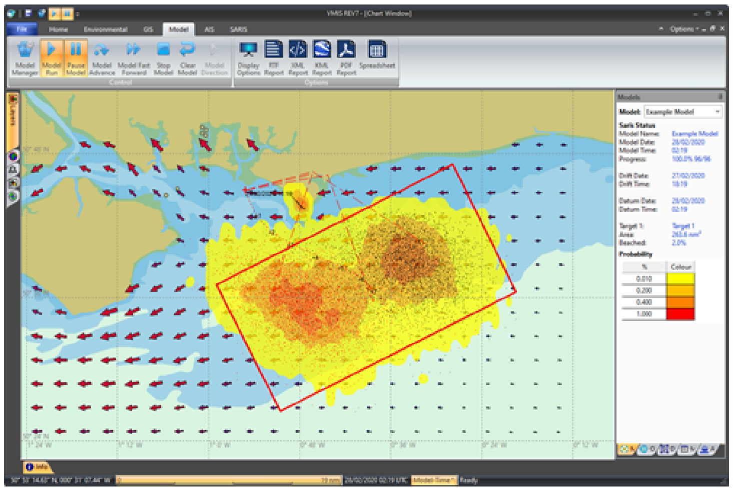


Figure 2.2: DroneResponse Interface

Another system used today is SARIS by (BMT, 1996). SARIS has been in use for over 20 years and is one of the market leaders within the maritime search and rescue interfaces. It has been in continuous development and is currently in version four with all the latest technology. SARIS is primarily focused on big ocean operations, however, it calculates the likely search area in which the target may be found from forecast oceanography data and meteorology charts and can therefore be used worldwide.



Model: Example Model

Saris Status

Model Name:	Example Model
Model Date:	28/02/2020
Model Time:	02:19
Progress:	100.0% 96/96
Drift Date:	27/02/2020
Drift Time:	18:19
Datum Date:	28/02/2020
Datum Time:	02:19
Target 1:	Target 1
Area:	263.6 nm²
Beached:	2.0%

Probability

%	Colour
0.010	🟨
0.200	🟧
0.400	🟧
1.000	🟥

Figure 2.3: BMT SARIS Interface

The key feature of SARIS is a clear interface, allowing the operators to make rapid setup during an emergency situation. As mentioned before, the Metocean forecast data is imported and used for the calculation of the search area as seen in Figure 2.3. It supports different planning facilities like Monte Carlo, Datum Point, Datum Line, and Backtrack. Besides calculating the search area, it also calculated the possible target trajectory from the forecast oceanography data and meteorology charts.

These interfaces provide examples of how predictive elements can be used to support rescuers in SAR operations. In the next section, academic research will be explored, which focuses on studying the effects of predictive elements in several different multi-UAV systems.

2.2 Predictive Interfaces

This section will present related works that have been focusing on studying different predictive and adaptive interface elements. This includes work that has focused on maintaining situational awareness in single operator multi-UAV systems, presenting the correct information to the operator at the right time, and comparisons between conventional and predictive interfaces regarding situational awareness and workload.

In (Roldán et al., 2019), a study on multiple UAVs with a single human operator was conducted with a focus on managing the high workload and maintaining the correct level of situational awareness. It is stated that conventional interfaces are not ready for this challenge, and new types of interfaces are needed for the task.

In (Roldán Gómez et al., 2017), four different multi-robot interfaces were implemented, and a study was conducted on how the situational awareness and workload were affected by each of the interfaces. Two interaction techniques were explored, including conventional desktop and virtual reality; both were developed with two variations, one with predictive and one without. Two predictive elements were studied; a way to show the user which UAV was most relevant to focus on and an element to show the user the current predicted risk of the UAVs. The predictive element was also integrated to automatically select the most relevant UAV instead of asking the operator to select it. The findings of this study suggest that predictive interfaces are not always helpful. However, the research problem of human swarm interfaces is still in an early phase, and automatically selecting the most relevant UAV might not be the path to follow.

In (Ilbeygi and Kangavari, 2018), a predictive single-human multi-robot remote system is implemented and evaluated. The predictive element of the system is called the intelligent adaptive interface (IAI) management module, which is in charge of delivering the right information to the user at the right time. This module is responsible for adapting the user interface given the current context. The findings of this study suggest that predictive interfaces significantly increased situational awareness, decreased the users' workload, and provided better mission completion times compared to the conventional interface.

In (Hou et al., 2011), the design challenges for an IAI are studied. The operator is still in control of the interface, however, the purpose of an IAI is to aid the operator in decision making and take specific responsibilities from the operator. The operator and IAI can work together and maximize the performance by providing the operator with decision-making aid that provides insight into current actions and what should be done next, increasing situational awareness. A hierarchical framework was proposed, where the IAI is split into three levels of agents. Senior level agents handle internal and external systems and manage the information flow and operations routines. Junior level agents acts on the input gathered from the operator, sensors, and overall environment. The working level agents then reason and model on all the information from the senior and junior agents and provide feedback for the operator.

It has been found that predictive interfaces can positively impact human factors and mission completion times and that conventional interfaces might not be ready for the challenges that arise when controlling multiple UAVs. There are inspiring examples of how interface design can include predictive elements to improve conventional interfaces in these research efforts. Yet, as previously mentioned, this area of research is still in its infancy.

2.3 Measuring and Understanding Human Factors

This section describes the essential human factors during monitoring of a SAR operation through a system interface. These types of missions with monitoring multiple UAVs may face several challenges that can lead to critical accidents. In Table 2.1 provided by (Roldán Gómez, 2018) the main human factors with the problems in such situations, with additional methods for detecting and solving the issues are listed. In the following sections, we provide more in-depth explanations for each.

Issue	**Problem**	**Detection**	**Solution**
Workload	Inefficiency and errors	Physiological signals Test (NASA-TLX)	Adjust autonomy Transfer functions
Situational Awareness	Inefficiency and errors	Actions and performance Test (SAGAT)	Immersive interface Filter information
Stress	Anxiety / boredom: Human errors	Physiological signals Test (NASA-TLX)	Adjust autonomy Filter information
Trust	Mistrust: Human errors Overtrust: Machine errors	Reactions Survey	Adjust autonomy Train operators

Table 2.1: Human factors in multi-robot missions (Roldán Gómez, 2018)

2.3.1 Workload

Task workload can be defined to be the subjective experience of the operator during the mission. In (Roldán Gómez, 2018), the workload is additionally described to be divided into three attributes: input load, operator effort, and work performance. Multiple dimensions of workload should also be considered, such as physical, mental, and temporal demands. In our case, it is expected that the operator would have to multitask, understand the situation, make decisions, etc. These factors will affect the mental workload and may cause errors in the operator's decision making and overall influence the mission performance.

In (Roldán Gómez et al., 2017) and (Roldán et al., 2019) the NASA-TLX assessment tool was used to measure study participants' workload. During the experiment, each participant was to be presented with two different interfaces. Before any of the testings began, each participant was asked to conduct the weighing step of the NASA-TLX assessment. After both interfaces had been evaluated, each participant was asked to conduct the scoring step of the NASA-TLX assessment for each of the two interfaces. The measured workload was a combination of all six of the NASA-TLX variables, but a scoring in each individual variable was also presented.

2.3.2 Situational Awareness

It is found that during a SAR operation that the situational awareness will affect the performance of the operator during the operation. Situational awareness can be defined as the perception of the elements in the current environment. In (hci901e20 et al., 2021) and (Roldán Gómez, 2018), it is suggested that the mental workload of the operator will affect situational awareness. Examples of this have a high workload will limit the available resources and therefore decrease the situational awareness. However, having a low workload will result in boredom and still decrease situational awareness. It is also noted that situational awareness is highly affected by the operator's workload. Having a very low and a very high workload can lead to drops in performance and SA.

Further research in (Endsley, 2004) states that SA is the ability for users to perceive, understand and make decisions in a situation fully. This includes three levels: *perception* (Level 1 SA), *comprehension* (Level 2 SA), and *projection* (Level 3 SA). Perception is the basic, only involving recognizing and monitoring elements. Whereas comprehension is built on perception and requires that the user can develop an understating of the situation through pattern recognition, evaluation, and interpretation of information. Lastly, being able to project future states of the environment is the highest level of SA. Further, the paper documents eight common design errors that affect SA and can be found in Section 3.1.5.

In (Roldán Gómez et al., 2017) and (Roldán et al., 2019) the SAGAT assessment tool was used to measure situational awareness during the experiments on different multi-robot interfaces. The SAGAT questionnaire had questions related to the past, present, and future locations and states of the UAVs. The questionnaire was given to the participants once in the middle of evaluating an interface and once more when an interface evaluation had been completed.

In addition to using SAGAT to measure situational awareness, (Ilbeygi and Kangavari, 2018) also used a questionnaire about the participant's overall perceived situational awareness, using a 7-Point Likert-type scale. This provided a more subjective measurement of the SAGAT questionnaire.

2.3.3 Stress

During a SAR operation, much responsibility is put on to the operator with their decision making and monitoring the operation. This may cause stress and impact the performance of the mission. In (Roldán Gómez, 2018) it states that neither high nor low levels of stress should be desirable, as low stress would result in boredom and high stress result in anxiety. Finding a balance in the level of stress will be the most optimal.

One technique to evaluate the state of a person's stress level is the subjective method NASA-TLX questionnaire related to the effort and frustration during the tasks. A more objective measurement would be monitoring the operator through readings of signals from the heart, skin, brain, etc.

2.3.4 Trust

Lastly, it is expected that the operator trusts the system, as relying on the system and its tools can be beneficial in an operator's decision making. However, having too much trust will result in overtrust that may cause the operator to rely too much on the system and causing mistakes that are not acted on. Too little trust will result in mistrust, and this can cause the operator to have too much responsibility and not using information that is provided, causing human errors.

In (Roldán Gómez, 2018) mentions two techniques for measuring the levels of trust. The first technique is observing the operator of his reactions during the usage of the system. Secondly, a survey can be given to the participant where questions about trust are given.

2.3.5 Related User Studies

Studies were conducted in (Jayam Patel and Pinciroli, 2021), (Roldán et al., 2019), and (Roldán Gómez et al., 2017) that used Situational Awareness, Task Workload, Trust, Quality of Interaction, Performance and Usability as metrics of measurement. The related user studies firstly base the experiment on a hypothesis. To give a context of the user study, the experimental design describes the scenario that the participants will be conducting the user study in. The participant sample is then described, which goes into detail about the recruited participants. Lastly, the procedures used during the user studies are discussed. All the metrics that are used for the participants during each task are then summarized. The results are then presented and analyzed in tables and graphs.

3 | Research Problem

This chapter discusses the findings of the related work. It reflects on the different solutions and findings that can be included in this project. Finally, the research gap is discovered and presented.

3.1 Limitations of Search and Rescue Interfaces

In this section, the features from the existing SAR interfaces are discussed and considered. While there has been work focusing on improving the effectiveness of SAR interfaces through improved user interface design, we now critically discuss these to build relevant features and functions to examine with our research prototype.

3.1.1 Probability Density Map

The existing interfaces SAROPS and SARIS both present a similar solution for visualizing the predicted zone of the system by using a probability density map. Since this method is used in both systems, it would be a viable element to explore by studying if the presented solution is the best or if some other alternative could present the information in a better way.

3.1.2 Weather Information

There are various ways to represent environment and weather conditions in SAR interfaces. Previous research and existing SAR interfaces seem to suggest projecting weather information onto the map. In the SARIS interface, water currents are displayed on the map. In comparison, the SAROPS interface allows for toggling interface elements on and off. Therefore, it should be considered to explore further into the user being able to control the visibility of weather information. The search area prediction in SARIS and SAROPS is based on environmental data, such as winds, tides, and tidal currents. Therefore, this information should also be available to the operator to entirely avoid relying on the systems area prediction.

3.1.3 Path Planning

Both SARIS and SAROPS display an optimal path expected to be used to search for the missing person during the rescue. By displaying the path that has been planned in the search onto the probability density map, it may affect the operator's awareness. This should be considered during the implementation.

3.1.4 DroneResponse

The DroneResponse interface from (Agrawal et al., 2020) provides several valuable insights into designing a multi-UAV system for aiding emergency responders. This section will highlight some of the most valuable findings related to our project and investigate where there might be some areas of improvement.

One of the aims of this project is to reduce the situational awareness of the operators. In their work, they discuss the SA demons by Endsley (Endsley, 2004), which are a set of eight common design errors that frequently occur when designing interfaces. In addition to the eight demons by Endsley, they also identify three additional demons from their own experiences of working with UAVs. Having some of these demons in mind when designing parts of the user interface could help reduce some of these errors and lead to an overall increase in situational awareness. The 11 demons will be described and related to our project in Section 3.1.5.

DroneResponse was created in a participatory design approach, in which they engaged in co-design with firefighters. This provided the authors with direct feedback on some of the design choices they made and changed some interface elements. Some of these changes could be taken into consideration when designing similar interface elements.

In this work, the designers provided various functions to support the SAR personnel; however, the search zone function is limited to a static representation. In waterways, the SAR operations are dynamic where the weather, current, and the victim may be moving. This is considered a limitation as the UAVs can also be searching unnecessary areas of the harbor.

3.1.5 Situational Awareness Demons

This section will describe the eight SA demons by Endsley (Endsley, 2004), as well as the three additional SA demons presented by (Agrawal et al., 2020). These SA demons will then be related to this project.

Attentional Tunneling is about how the user divides their attention. In complex systems, the user has to keep track of many aspects, both internally and externally. Attentional tunneling happens when the user is fixated on a specific aspect for too long, resulting in a loss of SA regarding the other parts of the system. In (Endsley, 2004) it is also noted that 35% of all SA errors fall into this category.

Attentional Tunneling is relevant to our project since the intended system will be introduced in an environment where the operator has many existing monitoring tasks. Therefore, designing the system to reduce attentional tunneling is essential.

Requisite Memory Trap is related to the limited short-term memory of the user and how much information from the system the user can keep track of. If the system relies too much on the user to memorize information, it can lead to SA loss as the user misses vital information.

Multi-UAV systems, like the one in our project, often contain much information. Keeping this SA demon in mind when designing the system and limiting the reliance on the users' memory could help improve SA.

Workload, Anxiety, Fatigue, and Other Stressors are all something that can harm SA. Some of these are not something that can be eliminated, like the stress of monitoring a SAR operation where lives are at stake.

This project will focus on the operator's workload, as the system should aid in the operators' current work and not impose more workload than before.

Data Overload is about the amount of data the human mind can process. If the amount of data presented to the user exceeds the user's capacity to process, it will lead to a loss of SA.

As our system has a large amount of data, this demon is relevant. It is necessary to design the system so that the user is not presented with too much information at the same time and let the system keep track of vital information to unburden the operator.

Misplaced Salience is about the elements that naturally draw the attention of the user. This could be bright flashing UI elements. Misusing these can lead the users' attention to the wrong elements and decrease the SA. While using this to highlight the correct elements can improve the SA.

Misplaces salience is essential to consider, as our system might have to draw the operator's attention at critical moments during a SAR operation.

Complexity Creep is related to the increasing complexity of today's systems. Complexity creep is when the user has trouble maintaining a precise mental model of how the system works because of the high number of features.

We will have to be considerate of this, as more features are added to the system.

Errant Mental Models are about the users' interpretation of information. Depending on the users' background and experience, they might have some mental model about how to process or view some specific data. Therefore, it is important to design systems that conform to these mental models.

Errant mental models are essential for our project, as the system is designed for professionals. Making sure the design of the system elements fits existing solutions could help alleviate this problem. This could especially be needed when designing and implementing the predictive elements.

Out-Of-The-Loop Syndrome is a side-effect of automation. Too much automation can cause the operator to lose attention to how the system and the automation perform, leading to a SA loss.

Since this project is about a semi-autonomous multi-UAV system, this demon is vital to consider. Having the operator lose focus at any point during the mission could be critical and could even result in the loss of life.

In addition to these eight demons from Endsley, (Agrawal et al., 2020) described three demons based on their experiences working with UAVs.

Transition failures across Graphical & Physical UIs is related to the errors that can occur when an operator changes from controlling the UAVs with a physical device to using a UI.

This is not something we will consider for this semester, as the focus is on the operations manager, who will not have access to any other control devices. In the future, it might be needed to consider when the systems for both the OM and IC are to be integrated.

Socio-Technical CPS Communication Failure relates to the problems that occur when a situation depends on a clear human to human, UAV to UAV, human to UAV, and UAV to human coordination.

When the two systems for the OM and IC are integrated, this is a demon that should be considered. All communication between the two operators must be received.

Enigmatic Autonomy is about the operator's understanding of the automation of the system. The operator needs to understand when and why a UAV takes the actions it does. For example, when a UAV returns to the base station to recharge.

This is a critical demon to consider in our system, as some UAVs might change their state during the operation, which needs to be evident to the operator.

3.2 Research Gap

While we have identified key areas in the state of the art research to the best of our knowledge, continued research should be made. We identify the most important research gaps that we will focus on in our research. This includes investigating predictive search zone presentation and its effects on situational awareness and investigating the predictive elements provided to the operators more closely in the SAR scenarios.

The work of DroneResponse in (Agrawal et al., 2020) is closely related to this study. They are focused on a system to support emergency response scenarios, such as fire surveillance and search and rescue, and collaborate with firefighters. The results of this paper document key design observations that emerged from the participatory design process to be used for other multi-UAV solutions. These design observations could benefit our project, but further study is needed, as the DroneResponse interface was tailor-made for the South Bend fire department and their rescue process. In contrast, our project is focused on the rescue process of Nordjyllands Beredskab, where the responsibility is divided among two key operators. Creating our project and prototype for a maritime search and rescue interface with different predictive elements enables domain experts to more clearly see the problem and provide us with feedback they could not otherwise have provided.

In (Roldán et al., 2019) it is stated that conventional interfaces for multiple UAVs with a single human operator are not ready for this specific challenge and that new types of interfaces are needed for this. Whether or not predictive elements impact the human factors within human swarm interfaces can be discussed. The results from the papers researched in the related works contradict each other. In (Roldán Gómez et al., 2017), no improvement on the human factors was found; however, the workload was increased with predictive elements. In (Ilbeygi and Kangavari, 2018), the predictive interface yielded a significant increase in situational awareness and a decrease in the users' workload.

Looking at the results and the predictive components of these papers, it has come to attention not to impose choices on the operator. The predictive element in (Roldán Gómez et al., 2017) automatically selects the UAV for the operator, which gave negative results in the human factors and specifically the workload increased. Another critique for the paper is that it focused on virtual reality, and overall, presenting four different interfaces in a user study is too broad. The conventional interfaces functioned as stepping stones to present virtual reality and highlight it. The predictive element of the conventional interface was presented as it was not carefully considered.

The predictive component in (Ilbeygi and Kangavari, 2018) shows that adapting and presenting the vital information to the operator provided positive results. They do not make the choices or actions for the operator but react to the context changes and present the vital information. Comparing the results against the results of (Roldán Gómez et al., 2017), this seems to be the way to go.

3.2.1 Predictive Search Zones

Several different predictive elements have been explored in the existing interfaces. However, more research can be conducted regarding predictive elements to show where the missing person might be.

The SAROPS interface, described in Section 2.1, uses a probability density map to show where the missing person could be, while the SARIS interface uses a heat map to show the most likely position of the missing person. The DroneSearch interface does not have any predictive elements for the search area, but the search area is still shown.

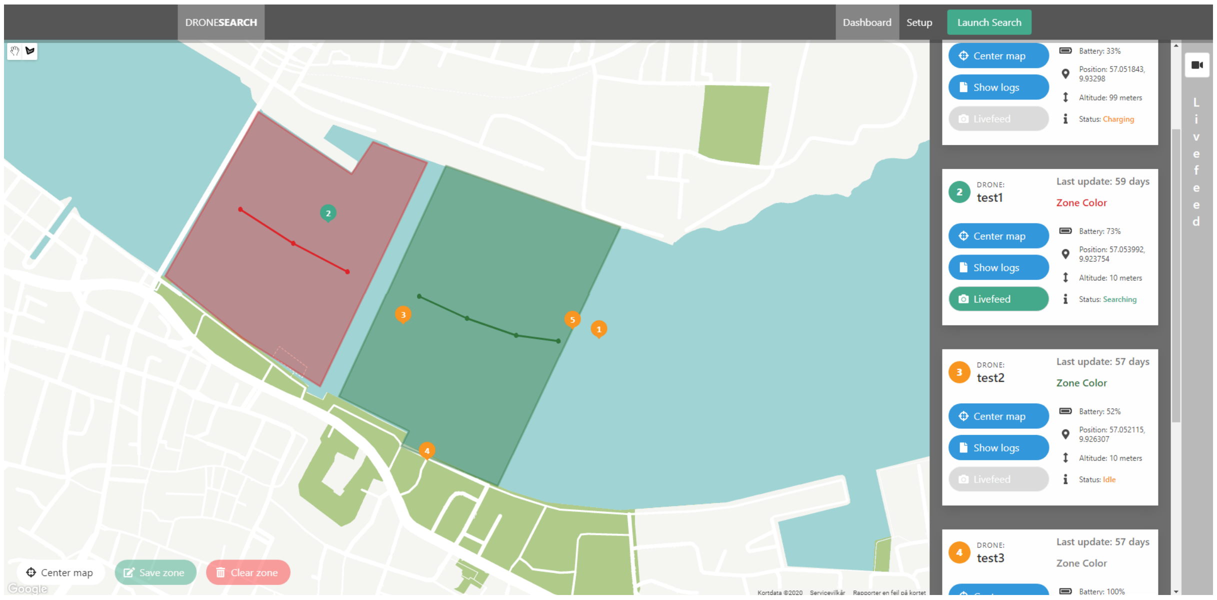
In all these interfaces, one similarity is that the search area is created by the system or statically defined in the system. One key difference to our system is that the operator is in charge of defining the search area from the provided information. Therefore, it would be interesting to explore how different predicted zones could aid the operator in defining this search area and how it could affect the operators' overall situational awareness and workload.

4 | Operations Manager User Interface: Previous Project

This chapter will describe the previous project and the implementation of missing requirements which emerged from expert reviews conducted. An overview of the previous project is presented before the implementation sections of the missing requirements.

4.1 Previous Project

The previous project (hci901e20 et al., 2021) was built as a research platform for studying future research questions. It is built for utilizing a swarm of UAVs that can communicate with a backend that processes and provides the data to a frontend. Each UAV is equipped with a Raspberry PI running an embedded system. The embedded system can control a UAV, create a mesh network, and communicate with the backend through socket communication. The backend can control and request information from each specific UAV through socket communication. A REST API is hosted on the backend connected with a SQL database and provides endpoints for retrieving the current state of the UAVs and sending commands to the embedded system on each UAV.



Battery: 33%
Position: 57.051843, 9.93298
Altitude: 99 meters
Status: Charging

DRONE: test1 (2)
Last update: 59 days
Zone Color
Battery: 73%
Position: 57.053992, 9.923754
Altitude: 10 meters
Status: Searching

DRONE: test2 (3)
Last update: 57 days
Zone Color
Battery: 52%
Position: 57.052115, 9.926307
Altitude: 10 meters
Status: Idle

DRONE: test3 (4)
Last update: 57 days
Zone Color
Battery: 100%

Figure 4.1: Web Application from previous project (hci901e20 et al., 2021)

A frontend, as seen in Figure 4.1 was created that provides an interface to an operator to interact with the system. The frontend requests data through the backend REST API and continuously retrieves the current states of all UAVs in the system. The interface provides an interactable map with all the UAVs plotted as seen in Figure 4.1. Additionally, detailed information on the UAVs can be found in the UAV overview that showcases the current position, battery, state, etc. The interface also supports drawing zones that can be sent to the backend to be processed. Additionally, the backend is used for controlling and handling all the UAVs, and the embedded system is used on each UAV as a communication link.

The backend uses the MVC pattern, and services were used to handle all calls through the Entity Framework to the database. The database includes two models, a model for a UAV and position. The models are in a one-to-one relationship, and for each UAV table, there is an associated position table. Communication with the backend consisted of a REST API used for the frontend and socket communication used for the embedded system. As seen in Table 4.1, there exist two types of controller endpoints, with a description of each of the methods.

Controller	**Method**	**Description**
Drone controller	GetDrones	Get all UAVs
	GetDrone	Get specific UAV from UUID
	PutDrone	Insert new UAV
	PostDrone	Update specific UAV
	DeleteDrone	Delete specific UAV
SearchController	LaunchDrones	Launch all UAVs
	AbortDrones	Recall all UAVs

Table 4.1: Backend Controller Methods from previous project (hci901e20 et al., 2021)

The socket communication in the backend and the embedded system will not be used for further implementation as these are unnecessary for the user study. It is expected that a demo service will be created that simulates all the UAVs and therefore replaces the need to use real UAVs. Further, an expert review was performed on two experts to gather feedback on the base platform. From the feedback, missing requirements were found, which is described in Section 4.2.

4.2 Implementing Missing Requirements

This section will describe the initial design and development process to implement the missing requirements from the 9th semester, which emerged from the two expert reviews of the previous semester's project.

The missing requirements from the 9th semester project, which has been selected to be implemented, can be seen in Description 2. An initial sprint will be conducted to implement each of these requirements, producing a platform with all the missing requirements for the last semesters' project. The design and implementation of each of the requirements will be presented in the sections below, where a more detailed explanation of the implementation can be found in Appendix A.1.

1. The interface shall not rely on scrolling, as it can make it difficult for the operator to find the relevant information
2. The interface shall show a precise alternative for GPS coordinates, as they are hard to communicate
3. The interface shall show points of interest on the map to increase the SA
4. The interface shall display UAV information next to the UAV's icon, such that operators do not need to search for the information
5. The interface shall support multiple monitors

Description 2: Selected Requirements

4.2.1 Scrolling

One of the problems mentioned during the previous semester's expert reviews was that the operator would have to scroll through the list of the cards used to present the information about each UAV, at least if there were more than 4-5 UAVs present in the system. Since the operator needs to have an overview of all the information available and the rescue operation itself, it can limit their ability to scroll in the interface to retrieve the information they need.

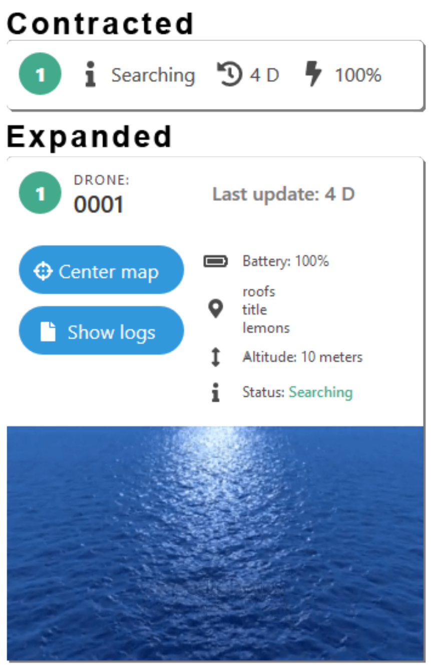


Figure 4.2: The two states of the UAV card component

An addition was also made to the UAV card component to improve flexibility for the operator and try to limit the need for scrolling. It now has two states, which can be toggled by clicking on the card. These two states can be seen in Figure 4.2. When the card is contracted, the status, time of the last update, and the battery charge can be seen. When the card is expanded, all the information about the UAV is available.

The need for scrolling is also further reduced by adding the UAV information on the map as seen in Section 4.2.4 as well as adding support for multiple monitors as seen in Section 4.2.5.

4.2.2 GPS Coordinates

Using the what3words technique mentioned in the hci901e20 et al. (2021) report as a precise alternative for GPS coordinates, it would be possible for the operator to communicate a GPS location direct communication. The purpose of this would be a faster way that is less prone for errors to radio communicate a location in a 3 meter square during a SAR operation. However, a limitation would be that the receiver would have to use the what3words tool to plot the location.

Implementing the what3words system is achieved by having a service with two methods, one for converting latitude and longitude to what3words format and a second for converting what3words to a latitude and longitude. As seen in Figure 4.2, the UAV card's expanded state shows the what3words information next to the GPS icon.

4.2.3 Points of Interest

Another thing mentioned in the expert reviews was the need for points of interest on the map. Points of interest are great for explaining a certain location through direct communication if needed and improves the operators' awareness of the map. The implementation was straightforward, as the Angular component used just needed another styling.

4.2.4 UAV Information on Map

Having the UAV markers on the map with numbers and a list with the corresponding information could cause confusion. It was suggested that a small info box next to all the UAVs could be implemented. This was implemented using an additional component within the component already used. The info boxes give the operator a better overview of the UAVs, as the information is right beside all the UAVs. Additional information can then be found in the sidebar or on another monitor explained in the following section 4.2.5. The info box can be seen on 4.3. Each info box can be closed by clicking the X in the top right hand corner of the box and opened again by clicking on the marker for the UAV.

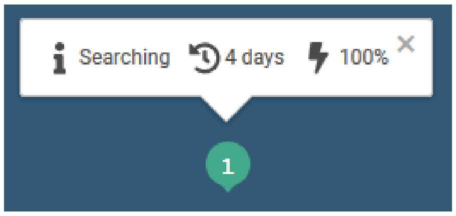


Figure 4.3: Information box above each UAV on the map

4.2.5 Multi Screen Support

As the operator will be in an office environment, it will be possible for the operator to use multiple screens to provide more detailed information about the SAR operation. To support multiple monitors in our system, we allow the operator to expand the UAV overview to a second window.

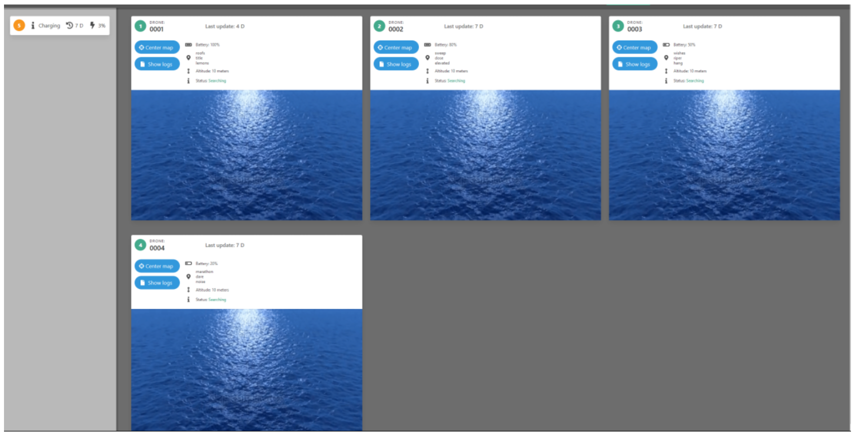


Figure 4.4: Multiple screens - UAV overview

As seen in Figure 4.4 the window will consist of a sidebar on the left and the UAV cards in the content on the right. The sidebar will list all UAVs in the system, and by clicking on the UAV card, the card will be transferred to the content where the operator will be able to see more detailed information about the UAV and the current live feed of the UAV when available. The operator would then be able to narrow down the UAVs that should show information. The windows can also communicate with each other. Thus, the operator can use the center map button on the individual UAVs, which would then center the map in the map overview window.

All the missing requirements gathered from the previous project have been implemented, and the system is now ready for further development, focusing on the role of Operations Manager.

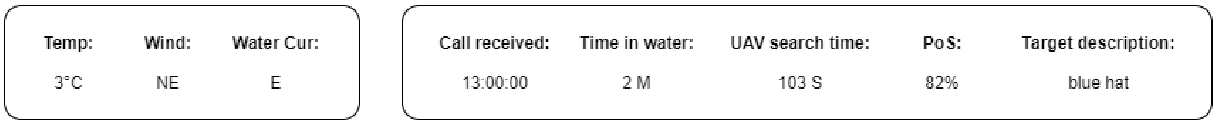
5 | Operations Manager User Interface: Design and Implementation

This chapter will describe the design and implementation of the prototype interface adapted to the Operations Manager, which is a further development on the previous project explained in Chapter 4. This prototype supports simulating SAR scenarios that can be used to evaluate the interface through user studies, which is also described in this chapter.

The final frontend design can be categorized into three parts; the header, the sidebar, and the map. There are no further implementations to the sidebar, meaning that there are no changes from Section 4.2. The header and map will be explained, and the chapter ends with an overview of the interface.

5.1 Header

This section will discuss the design and implementation of the header component. The header is at the top of the interface and displays mission-critical information.

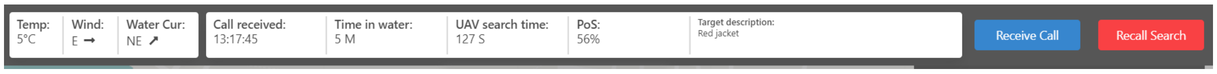


Temp:	Wind:	Water Cur:	Call received:	Time in water:	UAV search time:	PoS:	Target description:
3°C	NE	E	13:00:00	2 M	103 S	82%	blue hat

Figure 5.1: Prototype of Header Component

5.1.1 Design

The header should show essential data such as weather- and mission information. To the right of the information box, it should contain buttons for controlling the operation, one for receiving the call that instantiates a new SAR operation, and one for launching the UAVs. There should not be any elements overlapping the map, and placing this information and buttons in the header allows the map to be as large as possible. It also provides this information always to be available to be seen on the interface. The prototype of the header component is seen in Figure 5.1.



Temp:	Wind:	Water Cur:	Call received:	Time in water:	UAV search time:	PoS:	Target description:
5°C	E ➞	NE ↗	13:17:45	5 M	127 S	56%	Red jacket

Figure 5.2: Implementation of Header Component

As seen on 5.2 it was possible to utilize the top header bar and fit all the information needed, which enabled the map to be as large as possible. All the mission information and buttons are explained below and as to why they are shown.

5.1.2 Weather and Mission Information

The weather information displays the current temperature, wind direction, and water current. The temperature is displayed in Celsius and is shown because the current temperature significantly impacts the amount of time a person can survive. The wind and water direction arrows are shown to give the operator an indication of where the missing person can be if they are floating in the water. An important thing to note is that the arrows point in the direction, not where it is coming from. Further, the weather information is generalized for the specific area that the map is showcasing.

A collection of essential data received from the emergency call are displayed to the right of the mission information box. The specific time from when the call was received, how long the person has been in the water, how long the UAVs have been searching, a probability of success based on the temperature and duration in the water, and a description of the person missing.

5.1.3 Receive Call and Launch UAVs buttons

The receive call button is, as the name suggests, used when receiving a call and instantiates a new search and rescue operation. When clicked, a window opens with a form needed to be filled out, which contain essential information related to the emergency call. More detailed documentation on the implementation of the modal can be seen in Section A.3.2. After a zone has been drawn, it is possible to launch the search with the launch UAVs button. The launch is button is not available to be clicked before a search zone has been drawn.

The only slight difference from the prototype is the styling of the box and that the *target description* needed more space than the other information. However, overall is the header component implemented as expected and shows the same information planned in the prototype.

5.2 Map

This section covers the changes made to the map part of the interface. The map can be considered the most vital part of the interface, as the operator will get most of their information from this part of the interface. The map contains a large amount of information which is vital to the success of the mission. It is crucial to design the map elements so that they will not limit the operator's situational awareness or affect the mental workload by being too cluttered but still providing the needed information.

This section will cover the design and implementation of the following elements:

- Showing the rescue boat on the map, in section 5.2.1
- Additional information and action to the UAVs, in section 5.2.2
- Adding predictive elements for showing the missing person, in section 5.2.3
- Allowing the operator to toggle different UI elements, in section 5.2.4

5.2.1 Rescue Boat

This section covers the design and implementation of adding the rescue boat to the existing map.

Last semester's prototype provides an overview of the map of all the different UAVs. When looking at the system from both operators' point of view, it quickly becomes apparent that having the rescue boat itself as an element on the map is needed. Both for the IC, so they can orient themselves and their team in relation to the UAVs and the OM, so they know where the rescue team is and can direct them to any needed location.

The first part of this section will cover how the rescue boat element itself was designed and implemented, while the second part will cover how to visualize to the OM that the rescue boat is heading towards a found object.

Boat Marker

This section covers the design and implementation of the rescue boat marker on the map. This entails how to design the boat marker itself, but also how to show the rescue boat's field of view, as well as its current heading.

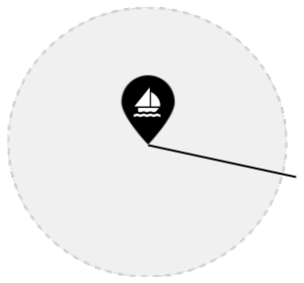


Figure 5.3: Rescue Boat Marker Mock-up

When designing this, one of the main ideas was to present the information about the rescue boat intuitively. There is a lot of elements present on the map, which could lead towards *attentional tunneling*, as described in Section 3.1.5. Therefore, the OM needs to quickly look at the boat marker to get all the information needed to not take too much attention from other elements in the interface. The rescue boat should contain the following three things: the boat's position, the heading of the boat, and the estimated field of view of the crew. A mock-up of how it was envisioned that this would look can be seen in Figure 5.3. Here the location of the boat is visualized much like the existing UAV markers, the heading is presented with a line pointing in the direction the boat is moving, and the FOV of the crew is visualized with a circle.

In order to support these features, a model needs to be defined for a boat object, which can be used in both the frontend and backend. The intended design of the boat model can be seen in Figure 5.4. The last variable in the model, namely *'EstimatedTime'*, is to be used to show how far away the boat is from a found object. This will be described in the next section.

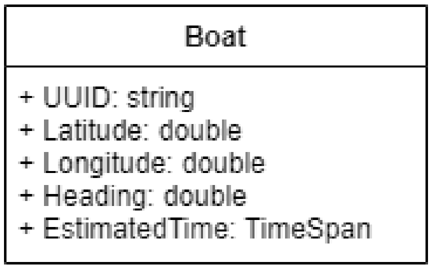


Figure 5.4: Boat Model

The detailed implementation of adding the boat to the map can be seen in Appendix A.2.3. To summarize, the boat model was created in both the frontend and backend. Since the backend uses Entity Framework for handling the database, the boat is created automatically from the defined model. In the frontend, a service was created to send a GET request to the backend continuously and retrieve the newest boat model from the database, which can then be shown on the map. In Figure 5.5 it can be seen how the implemented boat looks on the map.

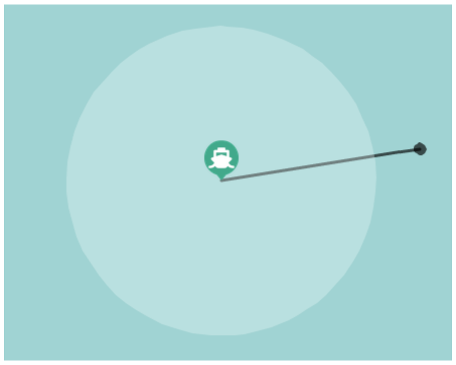


Figure 5.5: The Implemented Boat Marker

Estimated Time of Arrival

During a rescue operation, there will be critical moments where the two operators need to share information. One of these moments is when a UAV has found the missing person, and the rescue boat begins to move towards the person. Since both the IC and the OM have access to the system, it would be possible to share the information through the interface. In Figure 5.6 it can be seen how we envision this process will look. When a UAV finds an object and the OM has verified that it is a person, the IC will be notified. The IC will then accept the location and will begin moving towards the missing person. When the IC accepts, the OM should be shown that the boat is heading towards the person and how long the system estimates it will take.

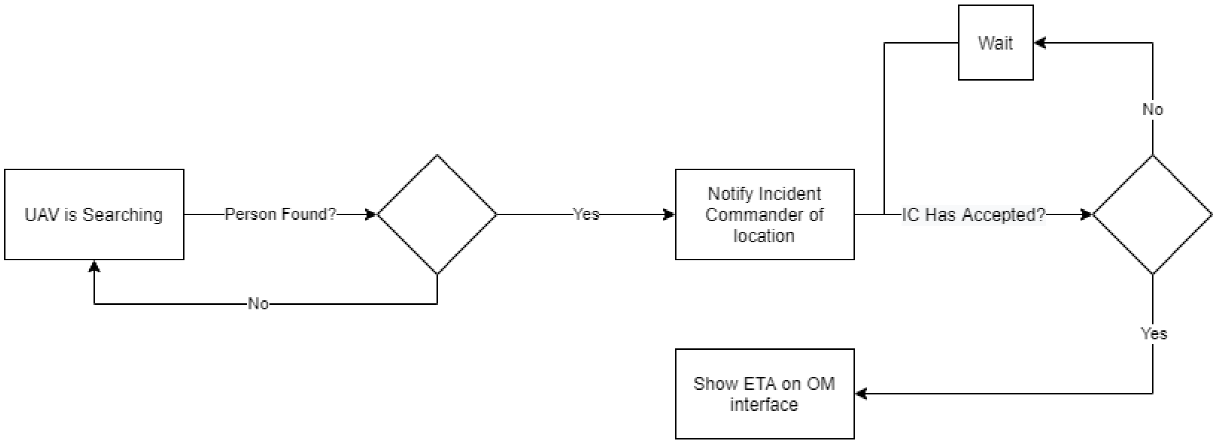


Figure 5.6: Boat ETA Process

Since the prototype developed this semester only focuses on the OM part of the interface, the part where the IC accepts and heads towards the missing person will be simulated. The detailed implementation of the ETA can be seen in Appendix A.3.6. To summarize how it was implemented, when a person has been found, the boat simulation is stopped, and a new simulation is started where the boat moves towards the missing person. A delay of five seconds is added here, to simulate the time it takes for the IC to accept the location. When the boat starts moving to the found person, a line is added between the person and the boat with the ETA shown below. In Figure 5.7, it can be seen how this looks.

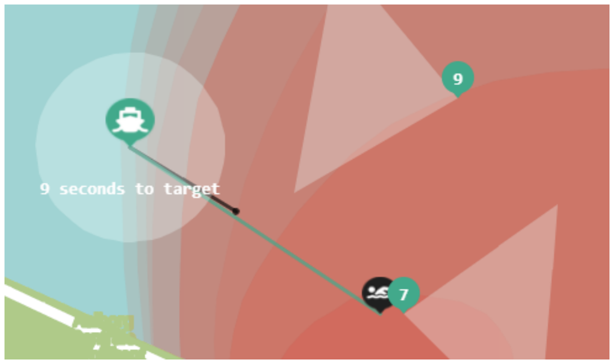


Figure 5.7: Boat direction and estimated time of arrival

5.2.2 UAV

Currently, the UAVs are represented with a marker on the map, each with their ID to distinguish them from each other. However, as similar solutions were explored in Section 2.1, it was clear that the UAVs needed to display more information on the map, besides the status color-coded marker.

FOV and Heading

Displaying the movement of the UAVs is currently nonexistent, their position is updated and shown by the new location. In DroneResponse, they show the UAVs' path by drawing trajectories behind each UAV, seen in Figure 2.2. This is also a viable solution with a few amount of UAVs, however this solution could be exposed to *data overload* explained in 3.1.5. Imagine 10-20 UAVs searching, and they all have the trajectories drawn behind them, it could be hard to know which trajectory belonged to which UAV. It is also hard to see where the UAVs are heading.

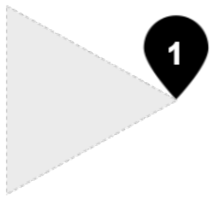


Figure 5.8: UAV FOV Prototype

The goal is to display as much information as possible most simply, so instead of showing the trajectory of each UAV, it was proposed to show the FOV. Showing the FOV also added the feature of seeing what each UAV could cover with their vision. A simple cone was proposed as seen in Figure 5.8. The actual implementation of the UAVs FOV can be seen on Figure 5.7. It is important to note that the current FOV cone is static and that it is set to a realistic width, however it can easily be adjusted to match real-world equipment and weather conditions. This implementation was to give the operator a sense of realism when using the system. The implementation of the FOV can be read in Section A.2.2.

Action Needed Window

The action needed window is displayed when the UAV detects an object in the water that the operator needs to be aware of. The idea is to show a video feed of the object and buttons to ask the operator if the UAV should follow or not. This window should be attention grabbing to the operator, and as seen in Figure 5.9, yellow is used for the headline. The buttons for yes and no have been placed as far apart as possible to reduce the risk of the operator pressing the wrong button, and they have also been color coded for visibility.

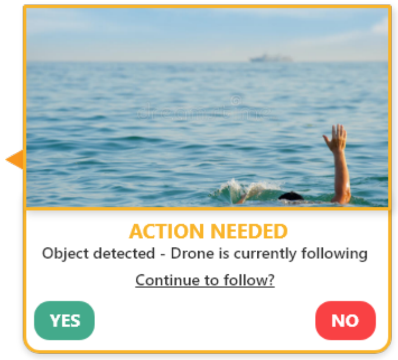


Figure 5.9: Action Needed Prototype

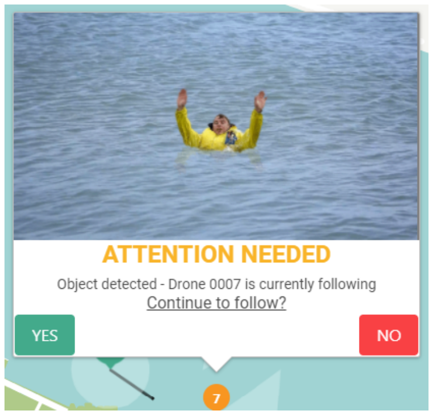


Figure 5.10: Action Needed Implementation

The actual implementation of the window is close to the prototype, as seen in Figure 5.10. Small changes have been made, the buttons have been made a bit bigger, also the headline has been made larger. The implementation of the notification box can be read in Section A.2.4. The text has been compressed a bit to make the video feed larger without compromising readability. The video feed is made as a moving GIF of either a bird or a person, and the implementation of this can be read in Section A.3.7.

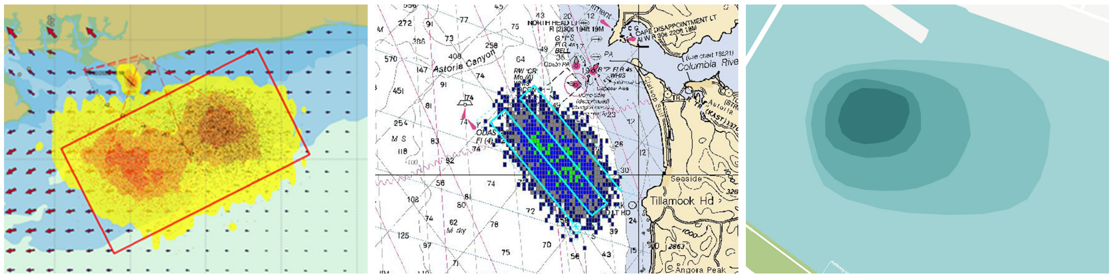
5.2.3 Predictive Elements

This section will describe the design and implementation of the predictive elements which will aid the operator in locating the missing person.

When looking at the two existing interfaces SAROPS (of Homeland Security, 2013) and SARIS (BMT, 1996), they both used a predictive element to show where a missing person might be. This predicted element was presented as a probability density map, where the zone was color-coded to show the system's confidence in where the person might be. Since this kind of prediction is used in two state of the art interfaces, this system will include a similar element. In order to try to compare how well this predictive element works, an alternative method of prediction was created as well. Both of these will be described below.

Probability Density Map

This section will describe the design choices that lead to the final look of the probability density map of the prototype. It was greatly inspired by the SARIS density map seen in Figure 5.11 (a) and the SAROPS density map seen in Figure 5.11 (b).



(a) SARIS **(b)** SAROPS **(c)** Mock-up of our heatmap

Figure 5.11: Comparisons of probability density maps

In both the SARIS and SAROPS interfaces, the main goal of the probability density map is to aid the operator in guiding rescue personnel to the correct location. However, in this system, the density map serves one other goal: to aid the operator in drawing a search area. This element will be refereed to as the prediction heatmap.

The mock-up of the heatmap for this system can be seen in Figure 5.11 (c). It has been made as intuitive as possible while still retaining important information. Both the SARIS and SAROPS density map can look cluttered, and a design like that would not fit in this system and might lead to data overload as described in Section 3.1.5. Keeping the heatmap simple allows it to not interfere with the other on-screen elements that the operator has to be aware of. As with the SARIS and SAROPS density maps, the mock-up is color-coded, signifying the probability of the person being in the area, where darker colors signify a more significant probability and lighter colors signifies a low probability.

The heatmap is represented in the system as a list of polygons, and the final implementation can be seen in Figure 5.12. Each polygon is added to the map and filled with a semi-transparent color. Layered more polygons on top of each other, then darken the color, showing where the probability is largest. The data for each heatmap is kept in the backend and can be retrieved through the *GetPrediction* endpoint seen in Table 5.1. More information about how the demo data is retrieved in the frontend can be seen in Section 5.3. Finally, the implementation of the heatmap can be seen in Appendix A.2.5.

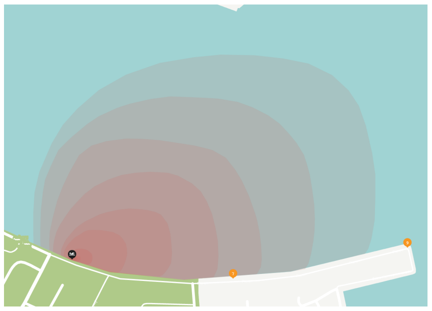


Figure 5.12: Implementation of the heatmap

Prediction Marker

This section will describe the design choices that lead to creating an alternative prediction method to the heatmap.

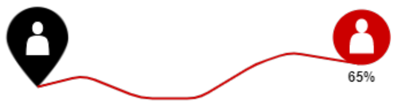


Figure 5.13: Mock-up of prediction marker

The main goal of this new predictive element is that it should aid the operator in drawing the search area, much like the heatmap, but should distinctly show the information. The heatmap provides the operator with an area prediction, where the system predicts the person's location. The idea with this new prediction element is that it will contrast the heatmap and provide more precise predictions of where the person could be.

In Figure 5.13 the mock-up of this predictive element can be seen. The idea is that a line is shown from the initial location of the missing person, shown by a black marker, to the predicted location, shown by a red marker. The red marker keeps moving as time progresses, and the line will then show the person's course. A percentage is added to the red marker, signifying how confident the system is in its prediction. This will be referred to as the prediction marker.

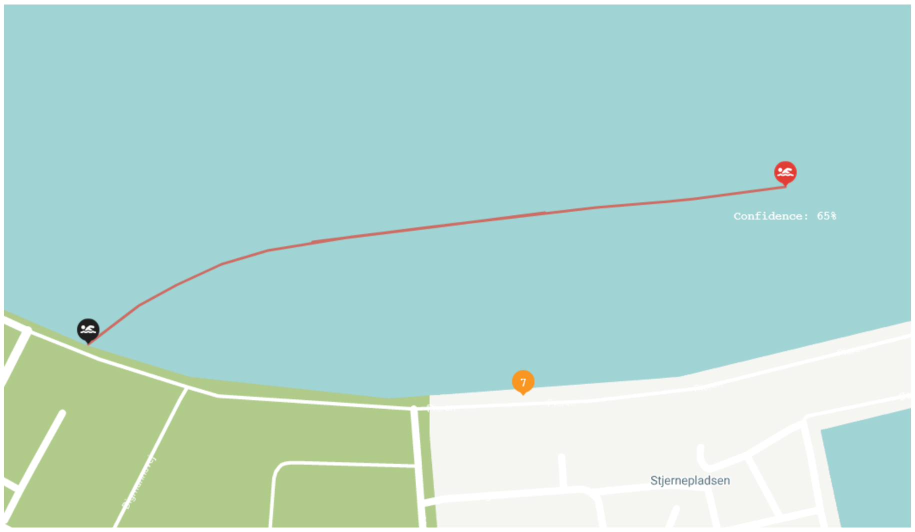


Figure 5.14: Implementation of prediction marker

The implemented prediction marker can be seen in Figure 5.14. The prediction marker is implemented much like the heatmap, but instead of a list of polygons, it will consist of a list of coordinates. For each coordinate, a confidence score is attached. This data is received in the frontend through the same endpoint as the heatmap. The actual implementation of the marker prediction can be seen in Appendix A.4.8.

5.2.4 Toggle Elements

This section explains the implementation of the UI box seen in Figure 5.15, which gives the operator the ability to toggle different UI elements in the interface. This was also a feature seen and used in SAROPS explained in Section 2.1. This interface is being designed for expert users, and they are expected to have preferences as to how their interface looks.

The idea of being able to toggle different elements is to reduce the cluttering of the interface. It can help reduce *attentional tunneling* and *data overload* as explained in Section 3.1.5. This is done through the operator being able to turn off different aspects of the interface and only focus on what they assess as the most important at the given time of the mission.

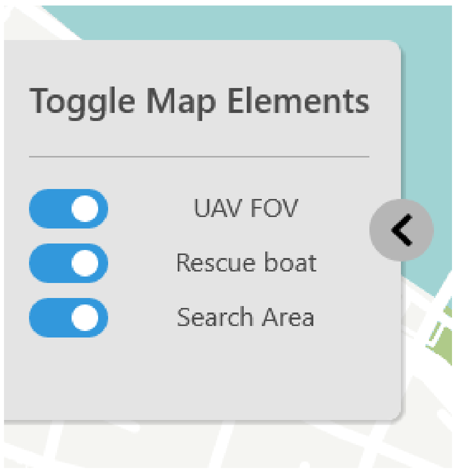


Figure 5.15: Toggle UI Elements Panel

As seen in Figure 5.15 it is possible to toggle UAV FOV, rescue boat, and search area. These elements cover most of the map, however, in SAROPS, it is possible to toggle almost every UI element to the operators' preference or needs. The toggle window has been implemented so that it is easy to add more elements so that if more elements are ever added to the UI, they will also be easy to add. The implementation of the UI toggle can be seen in Section A.4.4 which explains all the technical aspects.

5.3 Simulating Rescue Scenarios

This section will discuss the design and implementation of the demo simulation. The demo simulation was necessary for the user study, as it can not be expected that the study can be conducted during an ongoing SAR operation using deployed UAVs.

5.3.1 Designing Demo

The purpose of the demo simulation is to simulate an actual ongoing SAR scenario, where the UAVs are searching an area, with the rescue boat taking part in the mission to assist finding the missing person. Additionally, the demo simulation should support the predictive elements which are being explored in the study. The demo simulation should run different scenarios, as it is required that the participants use the interface with the different predictive elements. It is also expected that the participant should have a demo tutorial scenario.

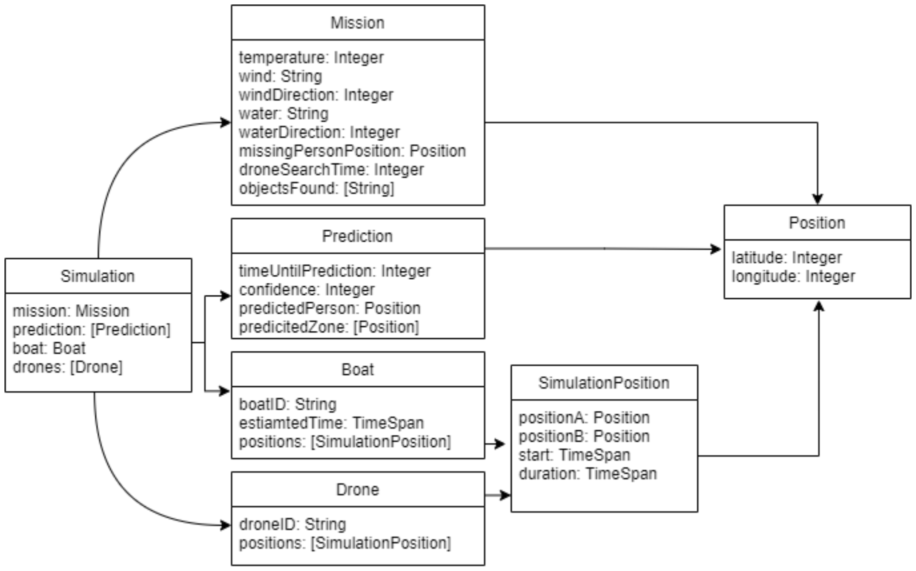


Figure 5.16: Demo JSON design

As seen in Figure 5.16 a proposed JSON design for the demo scenario is given. This figure showcases that it is expected that each demo simulation scenario, should consist of a mission, list of predictions, boat, and list of UAVs. The mission information contains data about the current scenario, such as the temperature, wind, and the last known position of the mission person. The predictions contain data about when the prediction should be shown, the confidence, and positions for either the predicted person or the predicted zone for the missing person. Further, the simulation should be able to move the rescue boat around during the scenario. To achieve this implementation, data of the boat ID should be given, and a list of simulation positions is a set of waypoints for the boat's movement. Lastly, the list of UAVs also has to be flying around during the scenario, and similarly with the boat, an ID of the UAV is given and a list of waypoints for the movement of the UAV.

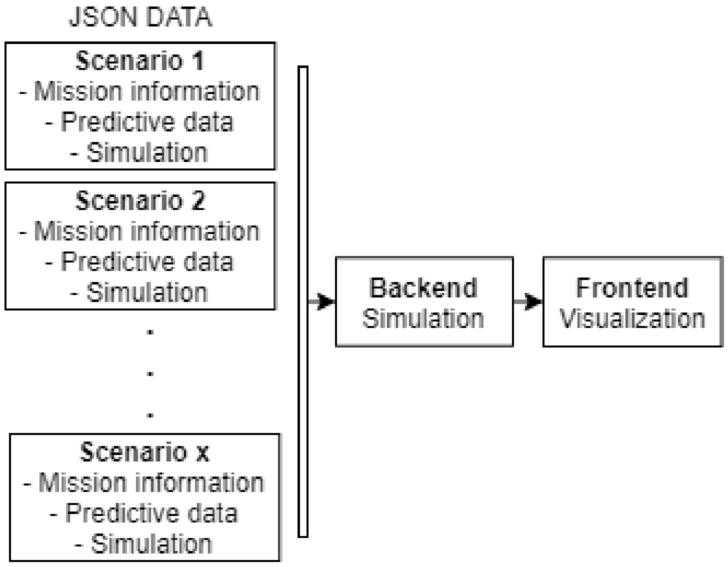


Figure 5.17: Demo design

In Figure 5.17 it is shown that multiple JSON files should be supported to create different scenarios. These JSON files should be handled in the backend to create the simulation that updates the UAVs in the database and allows for endpoints to receive this data. The frontend will then have to visualize the demo by receiving continuous updates from the backend while the simulation is running.

5.3.2 Implementing the Demo

By following the design, the implementation of the demo simulation consisted of implementations in both the frontend and backend. The sections below describes the decisions and how the implementation was developed will be discussed for both the frontend and backend.

Backend

To support the discussed design in Figure 5.16, models were created that represent the objects in the figure. With this in mind, the backend would be able to deserialize a JSON file into objects. With the deserialized JSON data used to compute the simulation, it would need to be represented into a simulation model containing the simulation data.

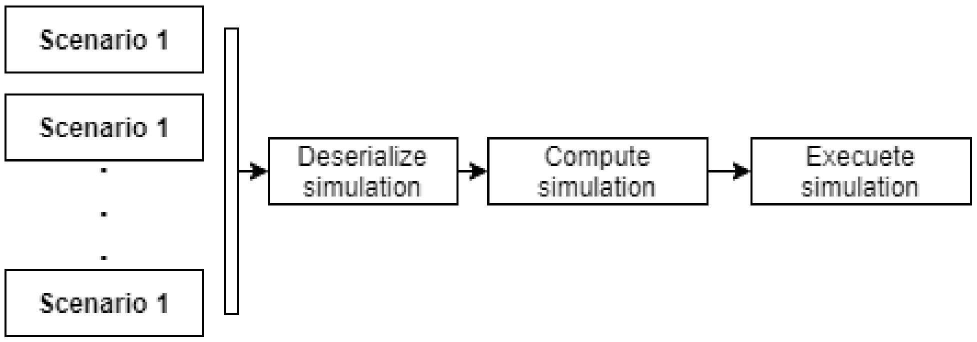


Figure 5.18: Backend demo

With the first step described in Figure 5.18, the implementations of the computations of the simulations were made. These computations were calculated from the instructions in the scenario JSON. The scenarios contain information as waypoints, and from that it is necessary to compute the coordinates between the waypoints to simulate the UAVs' movement. For this, a demo service was created, which handles everything relevant to simulating. Two methods in the service were created for computing the UAV simulation and boat simulation. These are used when going through the scenarios and computing the simulation objects for all the scenarios, which will later be used when executing the simulations. The simulation objects contain data of simulation execution time, the exact position of the UAV or boat, the battery, altitude, etc.

The last step in Figure 5.18 is the execution of the simulation. For this, two additional methods were created in the demo service to execute the simulation for the UAVs and boats. The execution relies on the data from the computation of the simulation. The implementation of the executing of the simulation is, therefore straightforward, and is implemented by going through the simulation objects and using the data to update the live data in the database. To ensure that the simulation is executed at the expected time, it keeps checking if the elapsed time is less than the execution time. Additionally, the execution of the simulation happens parallel on different threads. A further detailed explanation of the backend demo can be found in Appendix A.2.6.

Controller	HTTP Method	Method	Description
Demo controller	POST	LaunchDemo	Begins the selected simulation
	POST	PauseDemo	Pauses the simulation
	GET	IsPausedDemo	Returns boolean for demo paused
	POST	ForceUpdate	Forces update of scenario and prediction
	GET	IsUpdate	Returns updated scenario and prediction
	POST	SelectDemo	Selection of scenario in the simulation
	GET	GetPrediction	Returns prediction data
	POST	PostZone	Saves the drawn zone for a participant
	GET	GetMissionCompletion	Returns boolean for completetion of demo

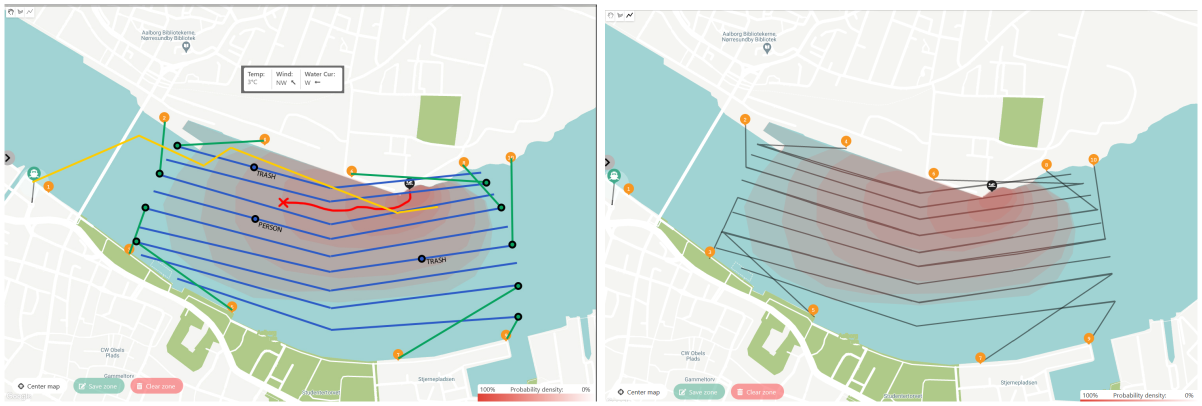
Table 5.1: Demo controller endpoints

A demo controller was created that supports endpoints for retrieving data and controlling the demo. These endpoints can be seen in Table 5.1, and further implementation details be found in sprint logs Appendix A.2 - A.4.

Frontend

As the frontend continuously fetches the UAVs from the backend, the simulation will be able to run and update live in the frontend. The endpoints in Table 5.1 are used to start the simulation when the UAVs are being launched. Additionally, in the admin panel in the frontend, the endpoints are used for controlling the demo and forcing the selection of scenarios and prediction.

Further, it was necessary to create the demo scenarios used by the backend to run the simulation. Creating the demo scenarios required tons of coordinates for both the boat and UAVs. The flying path also had to be a systematic flying pattern, where the UAVs would have to be parallel to each other. These coordinates were initially created by placing points on Google Maps and are described in Appendix A.3.5. However, this approach was very slow and unreliable, as the coordinates would have to be copied into the JSON file for each point.



(a) Preparing scenario **(b)** Drawing scenario

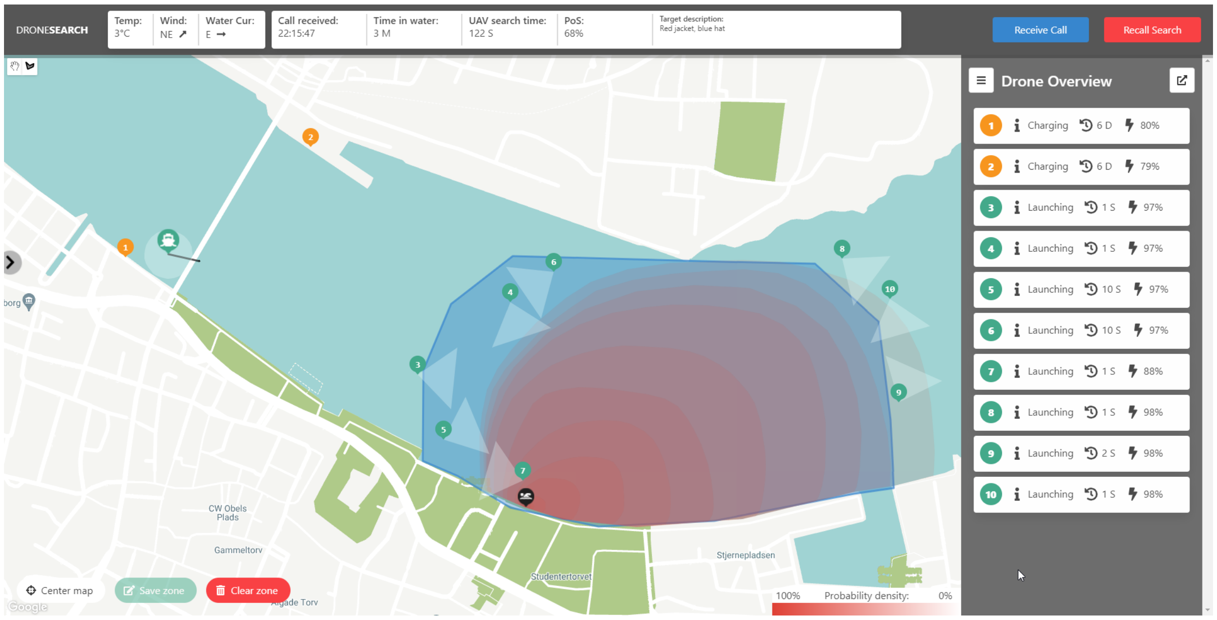
Figure 5.19: Representation of creating demo

It was figured that the frontend could be a great tool to draw the scenarios. By implementing a line tool in the frontend, it was possible to draw the flying patterns for each UAV, and ensure that they were flying parallel to each other, and then log the JSON object to the console to be used by the backend. This process saved much time and made it easy to adjust the scenarios. Further, the polygon tool was used to draw the zones used in the heat map prediction. The process of creating a scenario starts by taking a picture of the map by then discussing what kind of scenario that was going to take place, and with this it is possible to start to prepare the scenario.

As seen in Figure 5.19 (a), the relevant UAVs for the search in the scenario are decided, the green line shows the initial path to prepare the search, and the blue line is the search path of the UAVs. On the blue path, there are markers to show where objects are found and what type of object. Further, the yellow path is the rescue boat during the search. Last, the prediction is also given, with the heat map prediction shown as expanding polygons and line prediction as to the red marker. With the preparation done, the coordinates would have to be gathered of the same markers and paths. This was done in the frontend with the polygon and line drawing tool as seen in Figure 5.19 (b). The final scenarios can be found in Section 6.4 and a further detailed description of the creation of scenarios can be found in Appendix A.4.6.

5.4 Final System

The final system can provide predictions to help the OM draw zones and see mission information while following an ongoing scenario mission that provides all information on the map. On Figure 5.20 the final system can be seen. This gives an overview of all the components previously presented and discussed. The overview of the interface is taken when a mission is ongoing. The header information bar is filled out, and the search zone is drawn with the blue markings on the map on top of the heatmap, eight UAVs and the boat are searching for the missing person.


DRONESEARCH

Temp:	Wind:	Water Cur:	Call received:	Time in water:	UAV search time:	PoS:	Target description:
3°C	NE ↗	E →	22:15:47	3 M	122 S	68%	Red jacket, blue hat

Drone Overview

1	Charging	6 D	80%
2	Charging	6 D	79%
3	Launching	1 S	97%
4	Launching	1 S	97%
5	Launching	10 S	97%
6	Launching	10 S	97%
7	Launching	1 S	88%
8	Launching	1 S	98%
9	Launching	2 S	98%
10	Launching	1 S	98%

Figure 5.20: Overview of the final system

6 | User Study

This chapter will describe the process and results of the user study. The study is being held virtual with participants who have an affiliation within the UAV field. The goal is to measure the different predictive elements to document the differences using objective and subjective measures.

A pilot study was conducted to test the user study before actually conducting it with the actual participants. The pilot study was done to minimize errors and increase homogeneity. A total of **three scenarios** are conducted on **6 participants**, which are distributed using Latin Square.

6.1 Experimental Setup and Procedures

In this section, the setup and procedure of the user study are explained. Four different scenarios have been implemented, one tutorial and three scenarios to represent a real-life operation. There are in total 10 UAVs for the participant to control and monitor. The participant's goal is to follow the operation from receiving the emergency call to finding the missing person with the UAVs. Before the actual demo scenario start, they will have conducted the tutorial with a supervisor explaining the system. A more detailed explanation of each different scenario can be read in Section 6.4.

The user study is modified to be held virtually due to the constraints of the COVID-19 pandemic. Making the user study virtual has the benefit of making it easier for external participants to attend. The adjustments made to make the user study virtual can be read in Section 6.5. This also meant that the multi-screen support implemented in Section 4.2.5 was unused as the participant should share their screen. It was also not possible to require each participant to have multiple screens available when conducting the user study.

6.1.1 Participation Sample

A total of six participants were in the study. The participants were experts within different fields, some from human-computer interaction research departments, some from external companies with UAV experience, and some from UAV research departments. It should be noted that the participants are not Operations Managers but rather generally experts in the field with experience working with related interfaces and technology.

6.1.2 Latin Square

Each participant will be completing the scenarios in a random order to minimize the carryover effect. Another aspect is the fatigue effect, where the participants perform worse in the later conditions because they got tired or bored. For the distribution of the scenarios and predictions for each participant, the Latin Square is used. The Latin Square distribution ensures that each participant will complete each scenario once.

Order	1	2	3
Participant 1	Scenario 1 / Marker	Scenario 2 / Heatmap	Scenario 3 / Nothing
Participant 2	Scenario 3 / Heatmap	Scenario 1 / Nothing	Scenario 2 / Marker
Participant 3	Scenario 2 / Nothing	Scenario 3 / Marker	Scenario 1 / Heatmap
Participant 4	Scenario 1 / Marker	Scenario 2 / Heatmap	Scenario 3 / Nothing
Participant 5	Scenario 3 / Heatmap	Scenario 1 / Nothing	Scenario 2 / Marker
Participant 6	Scenario 2 / Nothing	Scenario 3 / Marker	Scenario 1 / Heatmap

Table 6.1: Participant Distribution

One of the challenges of distribution for the experiment was that it was needed to randomize two variables for both the three scenarios and the three types of predictions. The distribution ensures we get a good variation, but there might not be any significant statistical differences due to the low number of participants. The participant distribution for this user study can be seen in Figure 6.1. As there is three different variables, the scenario, the predictive element and the order in which the scenarios is presented to the participant, it gives 27 different combinations. However, due to being limited to a small number of participants, the combinations that can be studied is limited.

6.1.3 Procedure

First and foremost, the consent form will be signed, and then an introduction will be given as the participant completes the tutorial scenario. All the scenarios will run sequentially in the order specified by the Latin Square and will be paused midway as seen in Figure 6.1 for the participant to fulfill the SAGAT questionnaire. After each scenario, a NASA-TLX questionnaire will be given, and when the participants have completed all the scenarios, an interview review will be conducted.

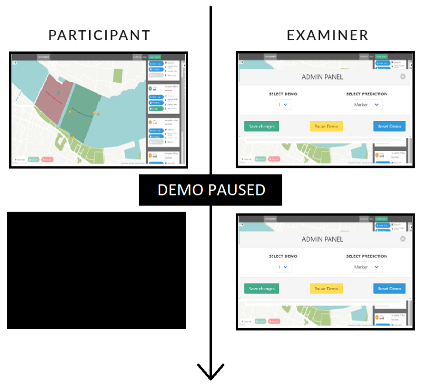


Figure 6.1: Pausing the demo

1. **Introduction (5m)**
   (a) The participant is introduced to the user study.
   (b) Consent form is signed by the participant and can be found in Appendix A.15.
2. **Tutorial (15m)**
   (a) Participant is read the introduction to the tutorial, which can be found in Appendix A.13.
   (b) Participant is guided through the process of receiving a call, drawing a search zone, launching the UAVs, confirming UAVs actions, and finally finding the missing person.
   (c) During the tutorial, the simulation is paused as seen in Figure 6.1, and the participant will be asked to fill out a SAGAT questionnaire to familiarize the participants with the stops and the SAGAT questions.
   (d) At the end of the tutorial, the participant will be asked to fill out the NASA TLX rating sheet to get familiarized with the process. After filling out the rating sheet, the participant will be asked to conduct the weighing step of NASA TLX.
3. **Scenario 1-3 (20m)**
   (a) Each user completes the given scenario assigned to them through the Latin Square distribution, seen in Section 6.1.2.
   (b) The scenario will be stopped once and the participant will be asked to fill out a SAGAT questionnaire.
   (c) At the end of each scenario, the participant will be asked to fill out the NASA TLX rating sheet. After the final scenario has been completed, the participants will also be asked to fill out the NASA TLX weighting sheet.
4. **Review / Interview (15m)** A semi-structured interview will follow the completion of the last scenario. Questions can be seen in Section 6.2.2.

6.2 Measures

This section follows a detailed description of the measurements used and how they are recorded during the study. We will include objective and subjective measures to understand how participants perform in the task using the interface and self-reported data that looks at the user experience.

6.2.1 Objective Measures

This section describes the objective measurements, which are measured consistently between the participants. The objective measurements include data of drawing the zone on the map in the different scenarios, where this data can be compared with the different predictive elements. The participants will also be given a SAGAT questionnaire during the experiment to understand the user's situational awareness.

Drawing Zone

One of the goals of having the predictive elements is to aid the operator in defining the search area. How well these predictive elements aid the operator is something that will be measured in two ways during the study:

1. **Comparing the drawn zones** When each of the participants saves the search area, the polygon is sent to the backend and saved in a file. This file allows us to compare each of the drawn zones after the study has been concluded. What will be most interesting is to see how well the zones fit the predicted areas, as this could provide some insight into how much each participant trusts the accuracy of each kind of prediction.

2. **Time spent drawing zone** A measurement of how long the participant has spent drawing the zone will be measured. This measurement will allow us to compare the different predictive elements and participants and examine if the predictive elements influence time spent drawing the zone.

3. **Distance from actual missing person** The distance from that the participants drawn zone will be from the actual missing person when found will also be measured. This measurement will be useful when seeing how the different predictive elements affects the distance from the target.

4. **Zone size** The area of the drawn zone will also be measured. These areas can be compared between the different prediction elements to show how much they differ.

Situation Awareness Global Assessment Technique

This section will describe the Situation Awareness Global Assessment Technique (SAGAT) and how it can be applied in this project's experimental procedure.

The goal of SAGAT is to get an objective measure of an operator's situational awareness. In (Endsley, 1988), it is mentioned that subjective techniques have been used before, where the operator is asked to rate their own SA. Providing a subjective rating has severe limitations, as the operator might not know what is going on in the environment. Endsley argues that the optimal situation would be to have direct access to the operator's mind and observe exactly what the operator knows. SAGAT is a way to get an idea of what the operator knows in a given situation. SAGAT is typically conducted on experts in a field (e.g., trained pilots), and it can therefore be expected that the operators know what to look for and where to keep their attention. Since this user study will not be conducted on rescue personnel, each participant will have to be introduced to what they have to look for. This introduction will be given to each participant before the start of the experiment. They will have a chance to practice using the SAGAT questionnaire during the tutorial part.

The idea is to ask the operators about their SA while they are in the environment, which is typically simulated. The questions asked are typically divided into three zones: immediate, intermediate, and long-range, which in order, describe their significance to the operator. Since SAGAT was initially designed for testing airplane systems, this often corresponds to how close elements are to the aircraft (Endsley, 1988). In Section 6.2.1 these three zones will be redefined to fit our UAV system better, and related SA questions will be formed.

Below, a brief walk-through of a typical experimental session with SAGAT will be described. The walk-through will be inspired by the description of the process in (Endsley, 1988) and rewritten to fit the role of a UAV operator instead of an aircraft pilot.

1. The participant will control and monitor the system in a simulated environment.
2. At a set time, the simulation is paused, and all simulation information is hidden from the participant.
3. The participant is asked a set of questions about their knowledge of the situation in the simulated environment.
4. When the simulation has been completed, the participant's answers are compared to the actual state of the simulation at the time of the stop. This comparison provides an objective measure of the SA.
5. A SAGAT score is determined for the system. The score is divided into three zones: immediate, intermediate, and long-range.
6. This process is repeated for several participants on the same system to gather enough data for statistical significance.

SA Questions

This section will describe the SAGAT questionnaire created for this experiment. The three zones of SA questions will also be redefined.

SA Question Zones Since the original SA zones are related to an aircraft and its technological limitations, they need to be redefined before a set of questions can be created. The redefined zones follow:

1. **Immediate** Elements that can potentially be life-critical. For example, elements that need the operators' input to divert UAVs to the missing person.
2. **Intermediate** Information about mission-critical elements. (Boat location, missing person prediction, etc.)
3. **Long-range** System information. (UAV states, locations, battery, etc.)

All the questions used for this experiment are documented and can be found in Appendix A.12.

6.2.2 Subjective Measures

This section describes the subjective measurements, which are measured from the participants' feedback. During the study, it will be relevant to measure the perceived workload of the users. We will use NASA TLX questionnaires, which will give a better understanding of the participants' perceived workload. Further, a review interview will be conducted on the participants. Both open-ended feedback can be given on how the system can be improved and feedback on which predictive elements they prefer.

NASA Task Load Index

This section will describe what the NASA Task Load Index (TLX) is and how it is used to be applied in the experimental procedure of this project.

NASA TLX is a subjective workload assessment tool, which can be used to get a measurement of an operator's subjective workload when working with human-machine interfaces (Gore, 2020). NASA TLX measures an overall workload score based on a weighted average of six different sub-scales (Gore, 2020): Mental Demand, Physical Demand, Temporal Demand, Performance, Effort, and Frustration. NASA TLX is administered through a standardized questionnaire, which has been made accessible from NASA. The questionnaire can be found in Appendix A.7 as well as the weighting table in Appendix A.8.

Experimental Procedure

This section will describe how the NASA TLX questionnaire is used in an experimental setting. This description is a summary of the experimental procedure found in the official manual (NASA), which is described as the usual sequence of events that is used with NASA TLX.

1. **Instructions** The participant is read the scale definitions and instructions. The manual has a standard set of instructions, but these can be modified if necessary.

2. **Familiarization** Participants practice using the rating scales after performing a small number of tasks.

3. **Ratings** Participants perform the experimental tasks and provide ratings on all six sub-scales. A rating sheet should be provided for each experimental task.

4. **Weights** Participants are asked to fill out the sources-of-workload comparison cards, which is a set of 15 different cards, each with two of the sub-scales shown. The participant is to circle the sub-scale on each card that they think had the largest influence on their workload. This step can be completed after each completed task or after a set of completed tasks, as long as those tasks are similar.

Data Analysis Procedure

This section will describe how the data from the experimental procedure is analyzed. This description is a summary of the data analysis procedure found in the official manual (NASA).

1. **Tally Sheet** For each participant, the "Sources-of-workload Tally Sheet" is used to calculate the weights for each sub-scale. The tally sheet can be found in Appendix A.9. To get the score for each sub-scale, it is counted how many times the participant circled that sub-scale during the weighing phase. The total for each sub-scale is then added to the "Weight" column when the counting has concluded.

2. **Worksheet** The "Weight" column from the tally sheet is transferred to the "Weighted Rating Worksheet," which can be found in Appendix A.10. Using this weight, an adjusted rating can be calculated by multiplying the raw rating with the weight. The adjusted ratings are then summed across the different sub-scales and finally divided by 15 to obtain the overall weighted workload score for the participant.

Introduction to Participants

This section will contain the two introductions presented to each participant when presented with each sheet for the first time. This introduction is needed to ensure that each participant understands how each sheet works, what the goal is, and knows the definitions of the six factors they are rating. The introductions are inspired by the two introductions provided by (NASA) (Section 6 & 7). The NASA TLX Rating and Weighting Sheet Introduction can be found in Appendix A.11.

Review

After the user study, a review interview will be conducted to gather more information about the participants' experience with the system. The interview consists of a semi-structured interview. This type of interview is chosen because it allows us to prepare the questions to guide the conversation and keep the participants on topic. It allows us to get more in-depth information about the participants' answers and provide us the reasons behind their answers. The questions used for this user study can be seen below.

1. Which of the predictive elements did you prefer the most during the demo?
   - (a) Why did you prefer that element?
   - (b) Did you feel that there was an advantage in having predictive elements vs. not having any?
   - (c) Did the prediction aid you in drawing the search area?
   - (d) How confident were you that the prediction was correct?
2. Did you feel that you were missing some information in the interface?
3. Did you feel that there was some unnecessary information in the interface?
4. What are your thoughts about the taking action dialog on the map?
5. Do you have any additional comments?

Other than the advantages explained above, it also provides us with qualitative data to compare the user study data. Conducting semi-structured interviews is a time-consuming task, however as explained in 6.1.1, this study is made with experts within the field, and getting detailed information about their reasoning is highly valuable.

6.3 Demo Tutorial

This section will discuss the initial demo tutorial that each participant will be presented. The demo tutorial aims to let the users become familiar with the interface and prevent factors such as inexperience in using the interface from influencing the user study results.

The demo tutorial is a 15-minute walk-through that presents all the essential features of the interfaces. These are the features that the user will be using and relying on during the user study. It begins with an overall introduction to the system and then goes through the essential features in each of the systems' three parts; sidebar, header, and map. Each feature is explained as well as how to operate the system. The introduction will be explained from a script to ensure that the explanations are consistent for all participants. During this, a predefined simple scenario will show all the aspects that can happen in a scenario. The list of instructions to present the demo tutorial can be found in Appendix A.13.

With these instructions presented in the demo tutorial, the user should now be familiar with the interface and its features. The participant is asked if they have any additional questions and are ready to begin the user study.

6.4 Search and Rescue Scenarios

During the user study, the participants will be presented with different mission scenarios to gather data on the effects of the different predictive elements. With this, there have been created **three different scenarios**, each of the scenarios can be modified to present no predictive element, the marker predictive element, or heat map predictive element. The scenarios were created to be as realistic as possible to simulate an actual mission. However, it should be noted that there is a limitation of the user study, that we do not have genuine data of these scenarios. However, the three scenarios were created to fit the typical rescue process, which Nordjyllands Beredskab described during several interviews. Three scenarios were created because this ensures the participants do not become too accustomed to one scenario and are described below.

Drunk person The first scenario is the drunk person scenario. This scenario is a call that is being received on the weekend at night, where it is reported that a drunk person has been walking along the harbor and went missing. The last seen location of the person was in front of *Musikkens hus* 5 minutes from when the call where received. He was seen wearing a blue jacket. The temperature is low, with only 3 degrees and a strong northeast wind and water currents going toward the east.

Suicidal person The second scenario is a suicidal person scenario. In this scenario, a call is being received that someone just witnessed a person with a yellow suit jumping from *Limfjordsbroen*. It is in the middle of the day, and the temperature is around 5 degrees, with both the wind and water currents going towards the east.

Fisher man The third scenario is the scenario of a fisherman falling off his boat. It has been spotted that a fisherman was suddenly missing from his boat. The last time the caller saw him was 2 minutes before he witnessed the empty boat. It was in the evening, and the temperature was around 3 degrees, with strong winds towards the northwest. The water currents in the water were going towards the west.

These scenarios will be used for all the participants in the user study and will examine how the human factors are influenced during the scenarios. The participant will be presented with the scenarios with the different predictive elements chosen from the Latin Square through the user study. This distribution will allow us to present all the different scenarios with all the different predictive elements and examine the results.

6.5 Pilot Study

This section describes the pilot study, which was conducted before the user evaluations took place. The feedback from the pilot study was mainly needed since the user study had to be conducted online due to the restrictions posed by the COVID-19 pandemic and to ascertain if the study or interface had any shortcomings. Therefore, the aim of conducting the pilot study was to answer the following:

- Is the introduction sufficient, and is everything covered?
- Are there any apparent shortcomings to the interface? Could anything be improved to make the evaluations easier?
- Is it possible to conduct the user evaluations online?

Since the system had to be evaluated in a remote setting, some changes had to be made to facilitate this. These changes can be seen in Appendix A.5. The pilot study was conducted with the group supervisor, who has not had any firsthand experience with the interface yet and was able to provide feedback on the introduction and the interface itself.

The study yielded some valuable feedback, which would have to be considered before the real user evaluations took place. Several suggestions regarding the interface were made. The suggestions and their implementations can be seen in Appendix A.15.1. Some feedback on the introduction was given as well, and these can be seen in Appendix A.15.2.

7 | User Study: Results

This chapter will present the results that were gathered from the user study. Section 7.1 will present the workload, situational awareness, and zone data results. The results of the reviews will be presented in Section 7.2. The participants completed the study without any major problems and provided excellent feedback on the overall interface. All the participants showed enthusiasm and felt very happy to participate and commented that they liked the interface.

7.1 Findings

The findings will present the results of the different measurements taken during the user study. The results of the NASA TLX questionnaires will be presented in Section 7.1.1, the results of the SAGAT questionnaire will be presented in Section 7.1.2, and the results of the zone drawings will be presented in Section 7.1.3.

7.1.1 Workload

The results of the NASA TLX questionnaires provided to each participant at the end of each scenario, and presents the workload of the participants. In Table 7.1 an overview of the total workload scores can be seen for each of the prediction types. At a glance, there is no significant change in the different workload scores across the three prediction types, which a maximum difference of less than one. Each of the six factors which lead to the overall workload score will be presented below, and a more in-depth view of the results will be presented.

Prediction Type	**Workload Score**
Marker	30.66
Heatmap	30.73
Nothing	29.86

Table 7.1: Summary of NASA TLX scores (lower is better)

The distribution of the overall workload score can be seen in Figure 7.1. The box plot shows the distribution of the workload scores across the three prediction types and the order in which the participants completed the scenarios. This graph shows that the workload for each of the three predictions does not vary too much. The median values for all three predictions are in the range of five to six. There also does not seem to be any significant differences in the score when looking at the order in which the participants completed the scenarios.

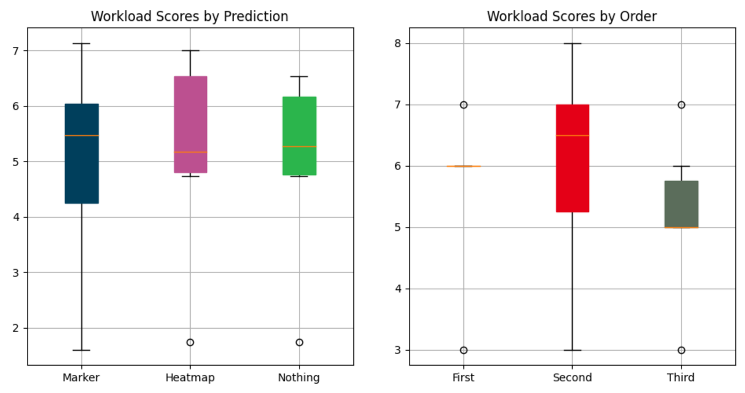


Figure 7.1: Workload scores by prediction type (lower is better)

The results can be analyzed further by looking at each source of workload for the different prediction types. These results can be seen in Figure 7.2. Here it should be noted that only five of the six sources are shown, as none of the participants scored any points in the *physical demand* source. When looking at each of the five sources individually, it is possible to see differences in scores between the three prediction types. The most notable results are with mental demand, performance, effort, and frustration level.

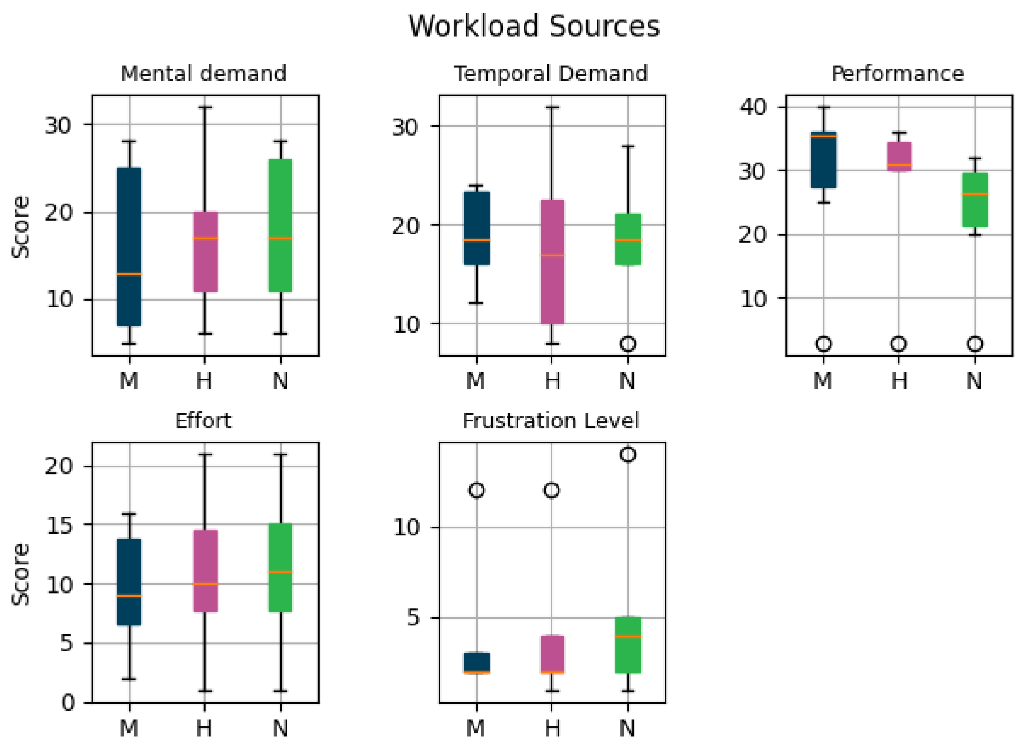


Figure 7.2: Sources of workload scores by prediction type (lower is better)

The mental demand is much lower for the participants when they are using the heatmap. Regarding performance, it seems that the participants felt their tasks were more successful when they were not using any prediction at all. The participants had the lowest levels of effort when using the marker prediction, followed by the heatmap, and the highest level of effort when no prediction was given. Regarding frustration, the participants were least frustrated with the marker and most frustrated without a predictive element.

7.1.2 Situational Awareness

This presents the results of the SAGAT questionnaires provided to each participant during each of the three scenarios, and presents the situational awareness of the participants. In Table 7.2 an overview of the total SAGAT scores can be seen for each of the prediction types. At a glance, the total scores are very close, but the scores for the participants when not having a predictive element was a bit better. The distribution of these scores will be described below.

Prediction Type	SAGAT Score
Marker	30
Heatmap	35
Nothing	36

Table 7.2: Summary of SAGAT scores (higher is better)

The distributions of the SAGAT scores can be seen in Figure 7.3 for the three different predictions, as well as for the order in which the participants completed the scenarios. Here it can be seen that the best results were from the scenarios where no predictive elements were used, followed by the predictive heatmap element. The results for the marker element were the lowest. However, it did not have any extreme outliers with low scores that can be seen for both the heatmap element and no element. In regards to the order, the score does not differ as participants completed more scenarios.

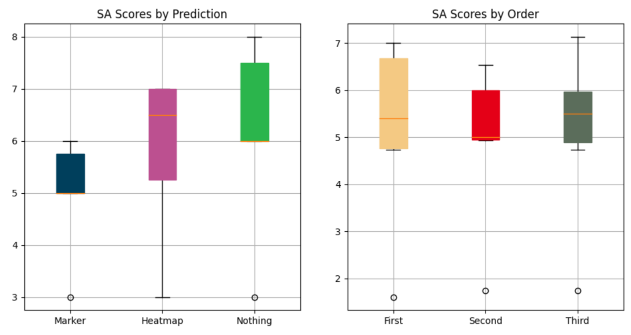
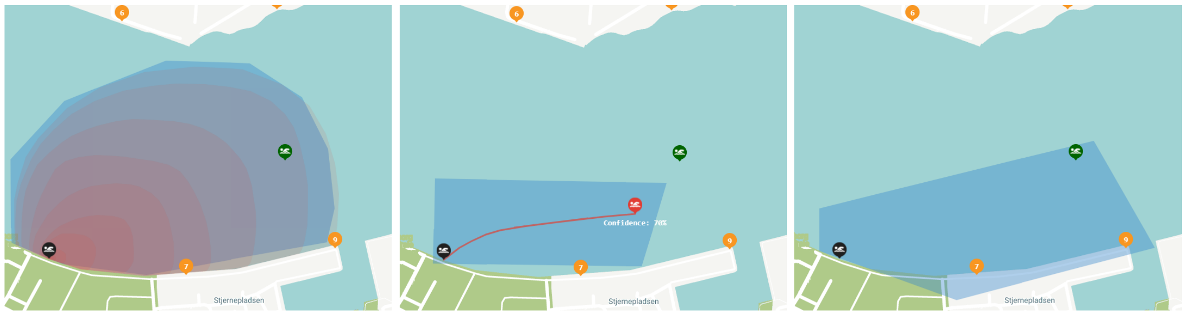


Figure 7.3: SAGAT scores by prediction type (higher is better)

7.1.3 Zone Data

This percents the zone data from the study and compares the participants' drawn zones throughout the different scenarios and predictive elements. The search zones of each of the participants were saved to our system for further analysis. These zones will be used to discuss how the participants used the prediction to aid them in drawing the zone and how much the participants trust the prediction. All the participants' drawn search zones can be found in Appendix A.16.



(a) Participant 3 - Heatmap - Order 3 **(b)** Participant 1 - Marker - Order 1 **(c)** Participant 2 - Nothing - Order 2

Figure 7.4: Scenario 1 - Prediction and search zones

Studying the zones drawn in scenario 1, which was presented to the participants in different orders as described in Section 6.1.2. The drawn zones seen in Figure 7.4 shows the search zone that the participant has drawn during scenario 1, as well as the given prediction that they were using during the scenario, and in which order they were presented the scenario. The blue zone is the participant's search zone, the black marker is the last know location of the missing person that the participant knows, and the green marker is the actual position of the missing person when found by the UAVs. By looking at the figure, it is seen that the search zone is influenced by the given prediction tool. Figure (a) shows that the participant mainly relied on the heatmap prediction to draw the zone. In contrast, in Figure (b), the participant marked the search zone around the marker prediction with a small area below and above. Lastly, in Figure (c), it can be seen how the participant performed when no predictive element was given, and they had to rely solely on the weather information to draw the zone. The other scenarios showed similar results.

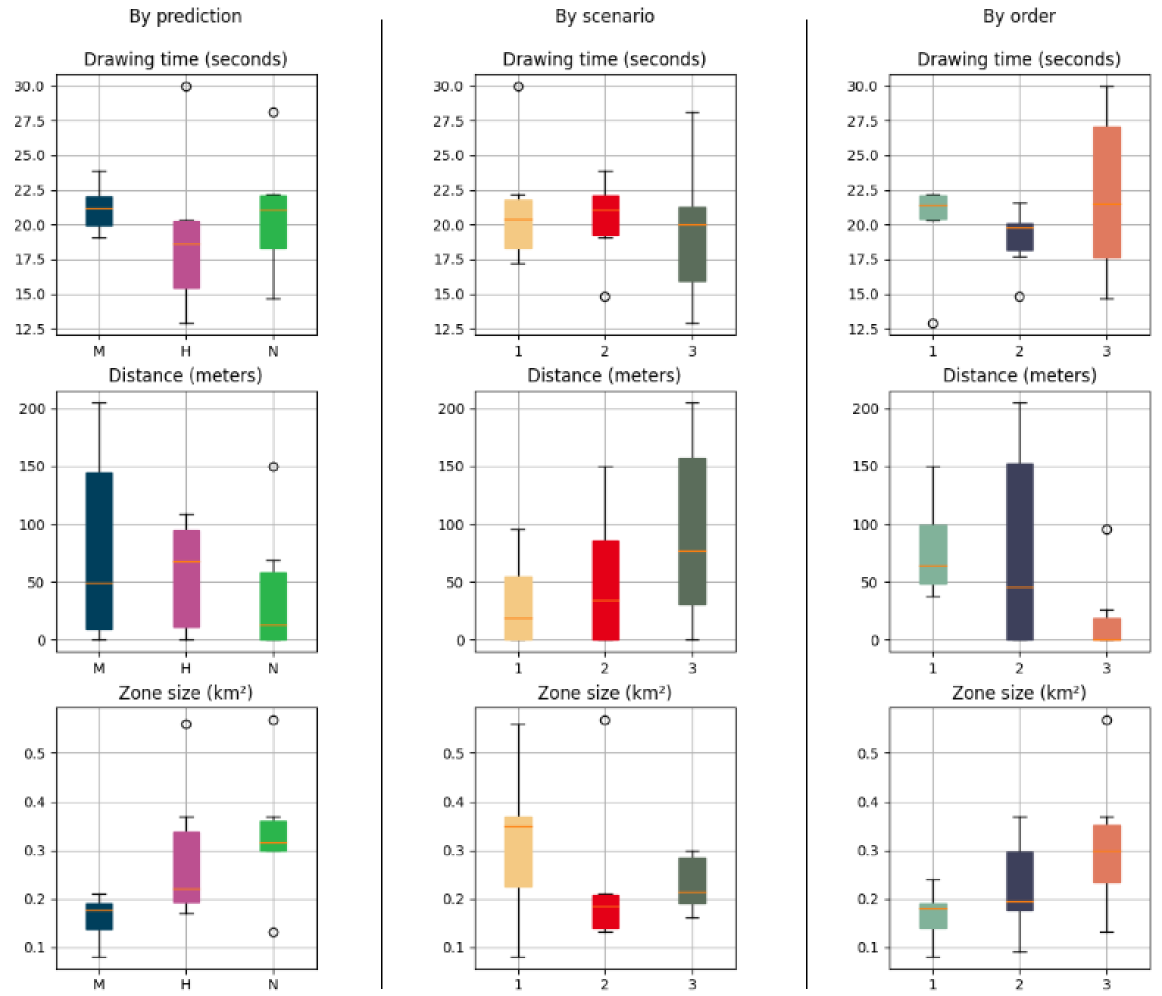
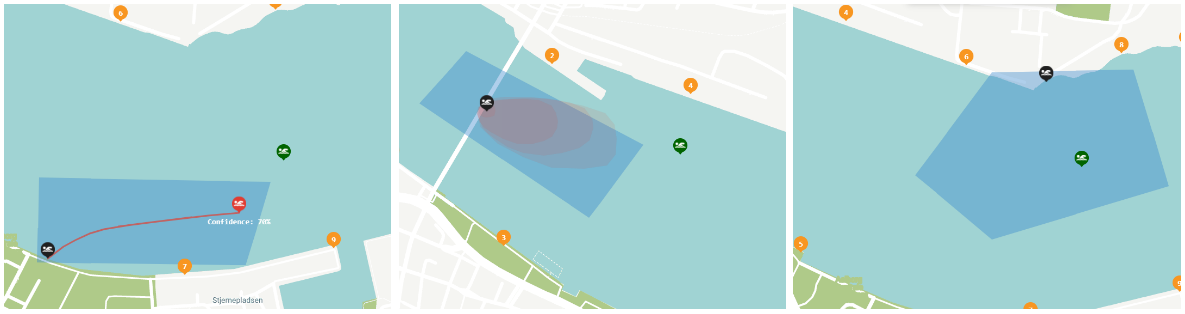


Figure 7.5: Participants zone data

In Figure 7.5 the participants search zone data is represented. This data shows that generally, the participants were faster to draw zones when there were the heatmap prediction element. However, the zones' size was generally bigger with no prediction, and thus the UAVs would have to cover a larger area. Examining the distances to the target, having no prediction generally provides the lowest distance, with the heatmap being similar. When the participants used the prediction marker, the range was between 0 to 200 meters from the target. Overall the participants struggled mostly with the distance from the target with scenario 3. One reason that might cause having nothing with the lowest distance to the target is that the participants would also draw the zones larger when having no prediction. Further, the data shows in by order, that as the participants use the system, their zone size was increasing.



(a) Order 1 - Scenario 1 - Marker **(b)** Order 2 - Scenario 2 - Heatmap **(c)** Order 3 - Scenario 3 - Nothing

Figure 7.6: Participant 1 - scenarios and prediction

Looking further into participant 1 search zones as seen in Figure 7.6 it is seen in Figure (a) and (b) how the participant is primarily relying on the prediction element to draw the search zone, whereas later when presented with scenario 3 with no predictive element the participant have to consider the weather data when drawing the zone. During scenarios 1 and 2, the participant had drawn a search zone that did not cover where the misusing person was found, as seen in Figures (a) and (b) with the green marker outside the zone. The search zone covers 0.3km², and by looking at Figure 7.5, it is also found that the zone is also the largest in scenario 3.

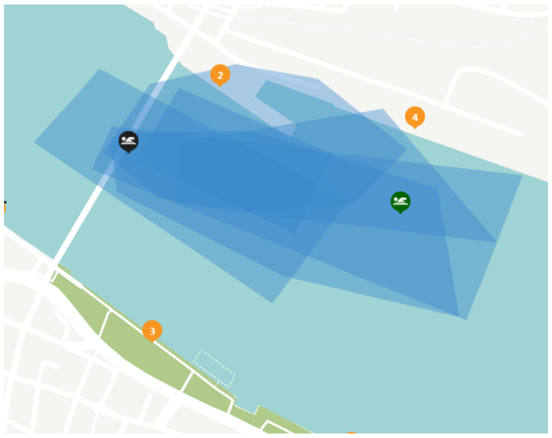


Figure 7.7: Participants zone data for scenario 2

Overall all the participants' search zones consider the weather data, such as the wind and water currents direction. As seen in Figure 7.7, the search zones in scenario 2 are generally heading towards the east, which matches the weather data.

7.2 Review

The interview was conducted on each participant after all the scenarios. All the participants are proficient experts within the UAV research and enterprise, and all the feedback is valuable for the interface. The interviews are summed up in this coherent section.

The overall feedback from the interviews was that the predictive elements helped because when there were none, they had to look at the weather conditions and use that information to draw the search zone. The prediction marker was really useful and gave a sense of probability without cluttering the screen with information. With the prediction heatmap, there was a slight confusion with it expanding too quickly because, at last, it was covering a large area of the water. However, a point made from a participant who preferred the heatmap is that the search zone is a 2D element, so the heatmap tells the area of where the person could be. With the prediction marker, the operator had to extrapolate the prediction percentage and draw around the prediction. The heatmap did the mental work for the operator, and the participant would draw a zone around that.

It was also mentioned that the heatmap fit the SAR operations better, as the object's position is unknown and the UAVs are performing a sweep search so that the UAVs are searching the heatmap thoroughly. The prediction marker can come off as imprecise when the UAVs are searching the whole area around the prediction. However, both methods should be introduced delicately to the experts so that it does not come off as how to do their job.

Another great feature was the *Action Needed Box*. It works well in two ways; one way in serves as a teaching mechanic for the operators to teach what the system potentially can do, and one way for database training through experience and expertise of the users to train the model to detect objects easier. There was a small translation issue where it is not clear what yes and no do. When presented with an action choice, the buttons should state precisely what is going to happen, so instead of yes and no, make them say *'Stay with the person'*, *'Continue the search'*. The user would then not have to make the mental connection between the question and the answer, especially in time critical situations. This is also seen in the *'Receive Call'* and *'Launch Drones'* buttons, which apply this idea.

A participant with expertise in monitoring UAVs suggested a new idea of how the data should be represented for each UAV in the sidebar. Instead of only showing the different elements' values, display a graph or some sort of graphical representation of the value. A graph is much easier to read and understand fast, rather than finding a number. It will also be possible to see how rapidly the battery capacity is going down or how the velocity of a specific UAV has changed. Another feature requested by the same participant was system logs. For example, having a yellow and red warning in the system, where yellow would be a warning that needs to be handled reasonably quick, a red warning could be a situation where a person's safety could be in danger. All the warnings would be stored in a log so that the operator can act on the warnings in the order that they find best.

A general critique received was that the buttons for drawing were too small and not intuitive. An idea proposed was that left click acts as the hand for moving the map around, and right-click acts as the tool for drawing the zone, which removes the need for the buttons. Another proposed idea was going from point and click, to drag and drop because the precision was not that important, especially to how the UAVs were searching. Thus having a big polygon that could be resized and dropped would be more than acceptable.

There was a slight disconnect in drawing the zone versus how the UAVs would search, and the UAVs flight pattern was confusing because they were lining up for a sweep search. However, after a couple of scenarios, it was noticed that they were doing a sweep search. A visual element was proposed to show where the UAVs were going so that when the UAVs are flying away from the drawn zone to line up for the sweep search, the operator would know what is going on. It might give added confidence in the system as a whole to see where they are going, like to have a path of each UAV drawn ahead of them.

There were missing details in the weather information, like how strong are the wind and water current. Some participants preferred the prediction marker because it somehow reflected how strong the weather currents were by expanding, where the heatmap failed to show this. Another feature to support this could be showing arrows on the map itself and not only show it in the header, which would work for both the different predictive methods.

Small changes like making the FOV of the boat bigger because it is surprisingly small and also make it so that it takes care of the weather conditions. If the weather is clear, it can probably see across the river. However, if it is cloudy, they can not see as far. Another point was that the UAV battery percentage is perhaps too much detail and could be split into ranges, such as low battery and fully charged. Further, if a person is missing, the operator is willing to let the UAV fall into the water because it is worth the risk of finding the person in the last possible second. It was even mentioned to remove most of the information in the sidebar and only have the status of each UAV as during the scenarios, it was not being used.

Interesting feedback received was regarding the procedure, specifically the SAGAT questions. It was pointed out that if all the data were like it was supposed to, it could be challenging to remember all the data. However, it could be interesting to check if the participant has noticed if some of the data were inconsistent or wrong. So placing incorrect data into the scenarios to check for the participants' awareness could be an addition to the SAGAT questionnaire.

The interviews gave a great insight into new ways to see and use the interface by experts within the field and, most importantly, gave valuable suggestions that will be discussed in Section 9.1.

8 | Discussion

This section discusses the results gathered from the participants through the user studies. Lastly, the limitations of this project include the final system, the procedure of the user study, and collaboration challenges.

8.1 Results

From the performed user studies on the participants, there was gathered much data on the participants' situational awareness, mental workload, searching zones, and feedback on the different elements. This data was presented in Chapter 7 and will be discussed in this section.

It was found that with the heatmap prediction that the participants had the largest mental workload score, with a minor difference from the others. However, investigating the different workload sources, it was found that the heatmap was generally less mentally demanding, and this might have been an effect of many of the participants solely relying on drawing the zone around the heatmap prediction area. It was also stated in the review that a participant would draw a zone around that. This means that as an operator, the system is expected to draw the zone rather than the operator and allow the operator to edit if needed manually. The heatmap also gave a higher score than the marker prediction with situational awareness. Further, it was found that the participants trusted the heatmap prediction and would often draw similar to the prediction. The trust meant that the drawing time for the heatmap prediction was the fastest. Some participants also preferred the heatmap rather than the marker prediction as it would be given an area rather than an exact location.

The marker prediction gave the participants the lowest levels of effort and frustration. Further, it gave the participants the lowest feeling of performance, which seems the opposite when looking into the participants' situational awareness when performing with the prediction marker, as it is the one with the lowest score. This data might be different if more participants were studied and with people more familiar with the system. With the prediction marker, the participants felt the same trust as the heatmap and would often draw the search zone around the path of the prediction marker. It was found that the prediction marker was the tool that required the most drawing time and with the largest range of distance from the actual found missing person. This result might also have been caused by the prediction marker being generally the smallest search zone, which means that the participant would have to be very specific for the zone to include the missing person. Participants felt that the marker was less cluttering on the screen, which may have positively affected the frustrations. However, it was also stated that the marker required more mental demand than heatmap as they would have to draw the search zone wider from the line.

Having no prediction element to aid in drawing the search zone showed the least overall mental workload score. By studying the different workload sources, it was found that the participants had the highest level of effort and frustration when having no prediction. The participants also felt the largest levels of performance. This could be caused by them feeling more achieving when drawing the zones as they have to focus on drawing the zones. The situational awareness results showed that having no prediction gave the highest level, with it being very close to the heatmap score. It might have been caused that they would have to pay more attention to the mission information to draw the zone. Feedback was also given on an alternative way to conduct the SAGAT questionnaire, where the participants would have to notice stuff that is not there. This idea might have altered the data as it would be more *'fair'* rather than the no prediction where the participant was forced to read the information to draw the zone. Last, the same principle with the other prediction would apply here, and the results would be better if more participants and experts were familiar with the system. The zone sizes were the largest with no prediction and were in the higher range of drawing time. Overall the participants preferred when they had a prediction element to aid them in drawing the zones, which gave them more confidence.

It was also found that the participants were drawing the sizes of the search zones larger for each scenario. This relation is likely caused by the carryover effect where the participant learns how to draw zones in the system and creates a larger zone to ensure that the UAVs cover enough to find the missing person inside the zone. Further, it was interesting how the participants would spend more time with their last scenario, and this could be caused as they wanted to perform better and spent more time thinking about their zones.

8.2 Limitations

A few limitations were found during the project, and some came from external factors, while others came from the choices made by the group. The limitations are discussed below.

8.2.1 System

The final system of the project is capable of simulating scenarios that were created beforehand, which was very useful when conducting the scenarios as these scenarios would give each of the participants the same experience with the system. Further, the system can also be linked up with real UAVs, as the previous project implemented an embedded system that could be hooked onto the UAVs. The frontend interface was also developed in Angular, which is a web application. This should be reconsidered to a desktop application, as it would give a more flexible system, such as the multiple screen support, which is very limited in a web application. The backend also used a database to store all the current positions of the objects, which were transmitted to the frontend through continuous GET requests. This was implemented in the previous semester's project. However, it would have been better to store the simulations in memory for faster handling of the system and store all logs into the database. This could have given a smoother simulation and might even have saved time during the implementation. Lastly, the continuous HTTP requests from the frontend to both fetch and post data were very ineffective. This should have been made into a WebSocket that would allow both the frontend and backend to communicate with live data.

As the scenarios are predefined, it limits the demo so that when the scenarios are running, most of the simulation is static. This means that when the participants are drawing zones, it will not influence the search path of the UAVs. The main focus is on how the participants will be drawing the zones from the prediction, and in that case, this limitation will not prevent the gathering of data on that area. Further, the speed of the UAVs and boats is hard to control, and it would have been relevant to have been able to control the speed of the objects with a variable instead of defining the time between two points.

8.2.2 User Study

The user study had some inherent challenges, as it had to be conducted online due to the ongoing COVID-19 pandemic. To limit the influence of any external factors, the setup of each session must be identical. Completely identical setups could not be achieved since each participant used their own office for the sessions. However, the procedure itself and each participant's tasks were kept identical and followed a script.

One could argue that this specific user study was especially well suited for virtual sessions, as the participants had to take on the role of Operations Manager, which would be located off-site during a rescue mission. This could have helped in improving the realism of the simulated scenarios. Multiple participants even mentioned that they felt the scenarios were realistic.

In terms of participants, the selection was beneficial in providing feedback to the system and interface, as they were experts in different fields related to HCI and UAV technology. The participants could confidently use the system after going through the training session but could not be considered expert users. It would have been better to also test this system on employees of Nordjyllands Beredskab over several sessions to test it on real expert users within SAR operations. This could also have helped improve the SAGAT questionnaire and maybe provide more varied responses to the different predictive elements. Multiple participants mentioned that they were not checking the information that were part of the SAGAT questionnaire but were trying to memorize them instead. As they felt that they had no reason to know that information during the scenarios as everything seemed to be working. Having real expert users would allow us to take measurements of actual situational awareness instead of memory.

An unforeseen advantage of having the SAGAT questionnaire in the middle of each scenario was that it could help keep the participants' attention. One participant mentioned that because they knew that these questions would be presented, they had to keep their attention on the interface at all times. This could help provide the participants with a sense of urgency, which otherwise would be difficult to obtain during a virtual demo.

8.2.3 Collaboration

Finding expert participants turned out to be a challenging task due to the ongoing COVID-19 pandemic. People were reluctant to meet in person and conduct the user study, hence why the user study was virtual. A total of six experts were found, and from that, much great feedback was received. However, more participants for gathering data would have been good since six is in the low end to show a statistical significance. Another issue with the participants is that none of them have experience with SAR operations. Optimally, a long-term collaboration with a company conducting SAR operations where the actual operators of the current solution would work with the interface and provide feedback related to issues with the current system.

Working with a company with experience in SAR operations could also bring realistic data for the scenarios to better reflect a real-life situation. Making the scenarios as real as possible was challenging, and some participants even mentioned that the heatmap zone expanded too quickly. The project group have no experience within real-life SAR operations and had to rely on common sense regarding the weather conditions and research. An aspect is also the length of each scenario because each participant should do a scenario with each of the two predictive methods and one without any prediction. The goal was to finish each user study in a reasonable time so that the participants easier can find the time for it, and for this an hour was chosen. The participants are experts with a busy workday, and a user study is hard to conduct in lesser time.

Another issue related to finding external participants not familiar with the interface is that the carryover effect might affect the results, in contrast to having trained experts within the interface that starts with the same knowledge. Conducting the user study virtual went well overall, but doing so also removed the implemented multi-screen support. Again, it was not expected that the participants had multiple screens, and therefore, the scenarios were designed for single screen operations. It was also impossible to control the environment. If it were conducted in a lab at the university, it would have been possible to control most of the external variables that can affect the user study. The participants were seated at home or in their offices, and some of them had background noise that could have affected their human factors while conducting the study.

8.3 Reflection

This project was carried out during the COVID-19 pandemic, as previously mentioned, which resulted in some challenges with the collaboration with external partners and user studies. There has been an extensive attempt to get in contact with the emergency services in the local area to the university to get a collaboration with a company that conducts SAR operations. This attempt has failed to get a response in both the 9th and 10th semester due to unknown reasons; speculations could be in lack of interest or simply due to being busy. Making the user studies virtual went a lot better than expected and was quite successful considering the circumstances. Loads of useful feedback were gathered to the contribution of the project.

The internal group work has not been affected by the pandemic because the group has experience working together remotely every day in the previous semesters. However, the group has been meeting every day, like they would have with a physical group room, just virtually with a VoIP application. This program allowed the group to share their screen for peer-programming and whiteboards to explain particular problems. There have been no constraints whatsoever working internally virtual for this group.

9 | Conclusion

Designing a multi-UAV system for aiding in maritime search and rescue operations provides several challenges, especially regarding human factors. Through the related work, it was found that research in the field of predictive swarm interfaces was still in its infancy, and it would therefore be the focus of this project.

In order to evaluate such a system, without having an actual swarm of UAVs operating, three realistic search and rescue scenarios were created to allow several participants to evaluate the system. These simulated scenarios proved to both work as an effective training tool as well as an effective platform for data gathering. Since the system is designed for an off-site operator, the participants would not have any indication whether the system was real or simulated, and the simulations proved to be reasonably realistic.

By taking inspiration from several existing search and rescue interfaces, two distinct predictive elements were created to assist the operator in defining the search area for the UAV swarm. The interface was also improved to accommodate the role of the Operation Manager better. The experts from the user study were positive about the predictive elements and the interface in general but had some ideas for improvements. These improvements are described in Section 9.1. Even though the participants were positive about the predictive elements, it is unclear if they provide a distinct advantage over having no predictive element, as there are both pros and cons.

- For the heatmap prediction, it was found that the users had the least mental demand and that the user would often trust the heatmap completely, leading to lower drawing times.
- For the marker prediction, it was found that the users had the lowest level of effort and frustration, but the drawing times were the largest.
- When having no prediction, the users had the highest level of frustration and effort. However, it gave the users the best feeling of performing well.

From the results, it is unclear whether a predictive element is better than no predictive element. The participants themselves preferred to have the predictive element and felt that it helped them draw the search area. There was no strong preference towards either the heatmap or the marker, as it was split evenly among the participants. However, it was suggested that a combination of the two might be the best solution, and this will be explored in Section 9.1.

9.1 Future Works

In this section, the future works of the project are presented. The future work was gathered from the feedback received from the participants.

Interface Improvements

The feedback gathered throughout the studies from the participants' expertise provided many ideas for future implementations to the interface.

One of the suggestions was that the weather information in the system should be more precise and be shown on the map. The participants had not enough information from the generalized weather data in the header. Further, when action was needed, the buttons were not direct enough and should state the action that would be applied. This technique should be applied to all the buttons in such a system, as they are often critical, and the users should not make the mental connection of the action every time they press a button that says yes, ok, no, etc.

It was also mentioned that the button for drawing a zone should be larger, as these are critical and should be easily accessible. However, it was also stated that the button to start drawing a zone should be removed and that the user should be able to left-click to interact with the map and right-click to start drawing the zone. In addition, for faster drawing, some predefined zones could be used that could be resized and drag, and dropped onto the map. The participants would often not care much about the shape of the zone, rather the area that should be searched.

It should also be considered to combine the two predictive elements, heatmap, and marker. The participants felt that they both had their advantages, with the heatmap giving a sense of the area that the missing person may be in, whereas the marker had less clutter and the lowest level of effort and frustration. The prediction elements could be toggle-able and allow the user to choose either to show a single prediction or both.

Further, it was also found that many of the participants were confused when starting the search of the UAVs, and to avoid this confusion, the launching of the UAVs should provide more information. This information should include how many of the UAVs will be launched and showing the path expected for each of the UAVs to make it clear how each specific UAV will search the area from the given zone.

Data that is represented in the system should also be shown more graphically. For example, showing the current battery percentage would only let the operator know of the current battery. However, by showing a graph of the battery of the specific UAV throughout the search, the operator would get a better overview of how the battery has depleted over time. This overview means that the operator can quickly get all the information needed by glancing at the graphical representation rather than looking at a number and interpreting it. This technique of showing the data graphically should be applied to all the data represented in the system, such as the altitude, UAV status, etc.

Lastly, to let the operator know what is critical and less critical, a log in the interface should be implemented. This log should provide the operator with all the relevant information needed to know, enabling the operator to act on this information. The log could be categorized into three sections; critical errors marked with red, less critical errors marked with yellow, and normal that provide information. Such as if the current battery of one of the UAVs is in the lower range, and it is expected that it might run out before completing the search, it should be put into the log as a less critical error. The operator would then act on this log and tell the UAV to recall to base or continue its search. A more critical error could be that a UAV is almost out of battery and that it will crash, the operator can take over manual control and ensure safety.

Realistic Rescue Scenarios

As the user studies were conducted with rescue scenarios created from our best knowledge from research, they may be missing necessary information. This data could be gathered from observing an actual rescue operation with an Operations Manager. Further, the user studies should be conducted in a more controlled environment that simulates the location of the operation manager. As there may be many factors that affect the data, this would give more reliable and valuable results.

Expert Users

The participants of the user studies were mostly HCI and UAV experts, which meant that they could primarily give expertise on their knowledge with UAV interfaces. However, by working with SAR personnel, they could provide feedback on how they handle the operations and what is necessary for a real-life SAR operation. The SAR personnel would also give better data when using the system, as they are already experienced in using similar systems and know what to look for and how they perform searches.

A | Appendix

A.1 Sprint 1

This section contains a log of the first sprint which took place from February 15th to February 19th. This sprint focuses on implementing the missing requirements from last semester and then achieve a state of the art conventional interface.

A.1.1 Sprint Planning

This section covers the sprint planning meeting held by the scrum team at the beginning of Sprint 1. The user stories selected for Sprint 1 is seen in Table A.1.

User story	Story points	System
As a user I do not want to rely on scrolling when viewing information about the drones	2	Frontend
As a user I would like UAV information displayed next to the map icons	5	Frontend
As a user I would like a precise alternative to GPS coordinates	1	Frontend
As a user I want the interface to support multiple monitors	5	Frontend
As a user I would like to see points of interest on the map	0	Frontend

Table A.1: Sprint 1: Selected User Stories

The total amount of story points needed to implement all user stories is 13, which the scrum team is confident can be achieved during one sprint.

A.1.2 Scrolling

This section describes the implementation to the UAV card component, to allow it to have two distinct states. These two states were needed to allow the UAV card to expand and contract when clicked by the user, to limit the amount of space each card used in the sidebar, and in turn, limit or eliminate scrolling by the user.

First it was required that the card component could keep track of what state it was in. For this a boolean *isExpanded* was created in the component. This boolean can then be toggled when the user clicks the card.

The *isExpanded* boolean was then used in an Angular template, to control what HTML to present to the user. This can be seen in Listing A.1.

```html
<div class="card" (click)="onCardClick()" [ngClass]="{'large': isLarge}">

  <div *ngIf="isExpanded; else contracted">
    <!-- EXPANDED CONTENT HERE -->
  </div>

  <ng-template #contracted>
    <!-- CONTRACTED CONTENT HERE -->
  </ng-template>
</div>
```

Listing A.1: The two states of the UAV card component

A.1.3 GPS Coordinates

This section presents the implementation of the what3words alternative for providing a location through communication. The what3words provides a a grid of 3 meter squares of the entire world that each are assigned three words. The implementation of this requires using what3words API for converting latitude and longitude into a what3word.

For this a what3words service is created in Angular that uses the what3words token and implements the two methods for converting.

```
public convertTo3wa(lat, lng) {
    return this.get(`convert-to-3wa?coordinates=${lat}%2C${lng}&
    key=${environment.api.w3wtoken}`);
}
```

Listing A.2: Conversion to what3words

The first method that can be seen in Listing A.2 is for converting latitude and longitude into what3words. By using the get method that is extended from the ApiBaseService a HTTP get request is sent and this returns a JSON response with the three words that describes that location.

```
public convertToCoordinates(word) {
    return this.get(`convert-to-coordinates?words=${word}&
    key=${environment.api.w3wtoken}`);
}
```

Listing A.3: Conversion to coordinates

Secondly the A.3 method is able to convert the what3words back into latitude and longitude. Using the get request again to the api endpoint a JSON response is returned with the latitude and longitude.

The drone card is then able to use the what3words service and use the convert to word method by using the current latitude and longitude of the drones, and then subscribe on the method to await the response. The response then contains the what3words that is then displayed in the drone card as seen on Figure A.1 in the red square.

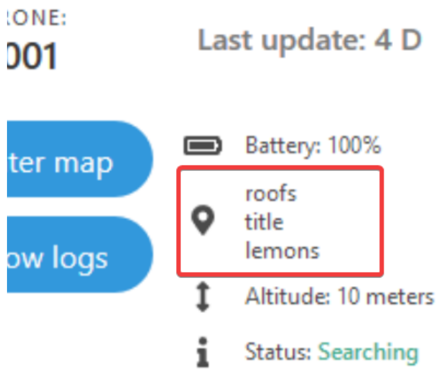


Figure A.1: What3words on card

A.1.4 Points of Interest

Adding points of interest on the map was an easy task, as reflected in the zero story points assigned in Table A.2. The component used called agm-map uses a JSON file for styling the map and in this file it is specified if points of interest should be visible or not.

```
{
    "featureType": "poi",
    "elementType": "labels.text",
    "stylers": [
      {
        "visibility": "on"
      }]
}
```

A.1.5 UAV Information on Map

It was needed to have some relevant information for each UAV be present on the map as well as in the UAV cards.

To implement this, an add-on component for Angular Google Maps was installed, called Snazzy Info Window. This package allows the placement of additional HTML elements along with the existing map elements. For each of the already existing markers, a Snazzy Info Window was added as well, with a simple Angular component that could render the most vital UAV information. This can be seen in Listing A.4.

```
1 <agm-marker *ngFor="let drone of droneService.droneList">
2   <agm-snazzy-info-window>
3     <ng-template>
4       <app-map-info-box
5         [inputDrone]="drone">
6       </app-map-info-box>
7     </ng-template>
8   </agm-snazzy-info-window>
9 </agm-marker>
```

Listing A.4: The Snazzy Info Window

As can be seen in the Listing on line 1, a agm-marker is created for each of the drones in the droneService. For each of these markers a agm-snazzy-info-window is created on line 2. Each of the snazzy info windows is then given an map-info-box component to render as seen on line 13. The map-info-box component is then given the drone as an input on line 14, which will be used to render the information.

A.1.6 Multi Screen Support

This section describes the implementation of the multi screen support added. First thing implemented was the button on the drone overview for opening a new tab with the drone overview page. As the new tab is opened, the drone overview on the dashboard is contracted as the information is now displayed on the new tab.

All the drones are displayed in a list with limited information on the left side and a grid view on the right side with all the information possible and the livefeed if active. When a drone is clicked on the left side it is moved to the right side. This is so that the operator can control which drones to focus on.

A problem that occurred was the communication between two different tabs and to establish communication between the windows an Angular package called Multi Window Service was used. This allows tabs to communicate and send messages between each others. This was useful for centering the map, here the drone ID are sent from the livefeed overview to the dashboard whenever the 'Center Map' button is clicked on a drone card. A lookup of the received drone ID can then be done in the dashboard window to then center the map, this lookup can be seen in Listing A.5.

```
private centerMapFromWindow(droneID: string) {
        let drone = this.droneService.droneList.filter(x => x.uuid == droneID)[0];

        this.mapService.centerMap(
          drone.currentPosition.latitude,
          drone.currentPosition.longitude
    );
}
```

Listing A.5: Lookup from drone ID

Each window subscribes to the multi window service and can then receive messages from the other windows. Show in Listing A.6 are the functions for sending messages implemented. These functions are implemented in the livefeed overview window, and can be used by that window. On line 6 the *sendMessage* function is called with the type *centerMap*, which is used later when receiving messages. The *sendMessage* function sends messages to all windows but itself, which can be seen on line 11. The multi window service is then used to send the message on line 12.

```
 1    ngOnInit(): void {
 2      this.multiWindowService.onWindows().subscribe(knowWindows => this.windows =
            knowWindows);
 3    }
 4
 5    public sendCenterMap(droneID: string) {
 6      this.sendMessage(droneID, "centerMap");
 7    }
 8
 9    private sendMessage(message: string, type: string) {
10      this.windows.forEach(window => {
11        if (!window['self']) {
12          this.multiWindowService.sendMessage(window.id, type, message);
13        }
14      });
15  pos   }
```

Listing A.6: Sending messages

Each window listens on messages from all the windows. When a message is sent to a specific window as described it will invoke the onMessage as seen on line 2 in Listing A.7. The message consist of data and the type of message, the type of message is used to find the specific method that should be executed for the type of message event. Currently the system only expects one type of message event which is the *centerMap*. The centerMap is then called with the message data as seen on line 4.

```
1 ngOnInit(): void {
2     this.multiWindowService.onMessage().subscribe((message: Message) => {
3       if (message.event == "centerMap") {
4         this.centerMapFromWindow(message.data);
5       }
6     });
7 }
```

Listing A.7: Reciving messages

A.1.7 Sprint Review

At the end of the first sprint, the Scrum team held a traditional sprint review meeting. In this meeting, it was discovered that all user stories planned in the sprint planning meeting was completed successfully.

User story	Story points	Completed
As a user I do not want to rely on scrolling when viewing information about the drones	2	✓
As a user I would like UAV information displayed next to the map icons	5	✓
As a user I would like a precise alternative to GPS coordinates	1	✓
As a user I want the interface to support multiple monitors	5	✓
As a user I would like to see points of interest on the map	0	✓

Table A.2: Sprint 1: Completed User Stories

A.1.8 Sprint Retrospective

In Table A.2 it is seen that all the selected user stories in Sprint 1 were completed, therefore, 13 user story points were completed during Sprint 1. In Sprint 2 the scrum team is expects that more than 13 story points can be completed during the sprint period.

A.2 Sprint 2

This section contains a log of the second sprint which took place from March 7th to March 15th. The sprint includes implementation of the necessary demo components, such as in the frontend predictive elements, and backend simulation service that allows for the demo to run.

A.2.1 Sprint Planning

This section covers the sprint planning meeting held by the scrum team at the beginning of Sprint 2. The user stories selected for Sprint 2 is seen in Table A.3.

User story	Story points	System
As a user I would like to see the FOV of the UAVs on the map	2	Frontend
As a user I would like to see the rescue boat on the map	1	Frontend
As a user I would like to see the systems prediction on where the missing person is	3	Frontend
As a user I would like to be notified when a UAV has found an object in the water	2	Frontend
As a developer I should be able to create a set of instructions for a demo simulation	5	Backend

Table A.3: Sprint 2: Selected User Stories

The total amount of story points needed to implement all user stories is 13, which the scrum team is confident can be achieved during this sprint like sprint 1.

A.2.2 UAV Field of View

This section describes the implementation of adding FOV cones to the UAVs on the map. This will aid the user in seeing which direction the UAVs are heading and/or looking.

The first thing that was needed, was to include a heading variable in the UAV models on both the frontend and backend. This variable will be used to draw the cone shape, and point it in the correct direction on the map. The heading is a double precision floating point in the range of 0 to 360, where 0 corresponds to north and 180 corresponds to south.

Next step is to calculate the three coordinates needed to create a triangle on the map, which can be used as the FOV cone. This calculation needs to happen every time the UAV is updated, and it will therefore be handled by the drone service of the frontend. The calculation will happen initially when loading all the UAVs, and after that, every time the position of a UAV changes. The code for calculating the FOV can be seen in Listing A.8.

```
1  private createDroneFOV(drone: Drone): Position[] {
2    let result = []
3    let center = new Position();
4    center.lat = drone.currentPosition.latitude;
5    center.lng = drone.currentPosition.longitude;
6    result.push(center);
7
8    // Calculate two edge points of FOV triangle.
9    result.push(this.moveInDirection(center.lat, center.lng,
         drone.currentPosition.heading + 40));
10   result.push(this.moveInDirection(center.lat, center.lng,
         drone.currentPosition.heading - 40));
11   return result;
12 }
```

Listing A.8: Calculating UAV FOV

On line 2 - 6 the current position is added as the starting point of the triangle. The two remaining points are then added on line 9 and 10 using the function *moveInDirection*. This function takes a latitude and longitude and a heading, and returns a new coordinate 100 meters from the initial point, in the given heading. To create the FOV cone this function is then used twice, with 40 degrees added and subtracted to the current heading of the UAV. In the end this list of three positions is returned and saved on the UAV model.

The final step is to draw the triangle on the map. For this an agm-polygon is added to the HTML for each of the drones, with the three positions given as input.

A.2.3 Rescue Boat on Map

This section describes the implementation of the user story: *As a user I would like to see the rescue boat on the map*.

Backend

The first step to implement this, is to create a model for the rescue boat. The implemented model in the backend can be seen in Listing A.9. The boat will contain a latitude, longitude, and a heading, which are the only three variables needed to display it on the map.

```
public class Boat
{
    [Key]
    public string UUID { get; set; }
    public double Latitude { get; set; }
    public double Longitude { get; set; }
    public double Heading { get; set; }
}
```

Listing A.9: Boat model in the backend

Now that the model has been created, a service and a controller is needed to get the saved boat model from the database. The **BoatService** currently has one function, called **GetBoat**, which can be seen in Listing A.10.

```
1  public async Task<ActionResult<Boat>> GetBoat()
2  {
3      using (var scope = _scopeFactory.CreateScope())
4      {
5          var dbContext = scope.ServiceProvider.GetRequiredService<DatabaseContext>();
6
7          var boats = dbContext.Boats;
8          var boat = await boats.FirstOrDefaultAsync();
9
10         return boat;
11     }
12 }
```

Listing A.10: BoatService - Function to return the boat from the database

Since there will only ever be one boat in the database, the function is rather simple. First a collection of the Boats database table is created on line 7, then the single boat object is found on line 8, by taking the first item of the collection. This boat is then returned.

Now that a service has been created which can get the boat object from the database, a controller is needed to provide this data to the frontend. The controller is created with the endpoint **api/boat**, and currently only contains a GET request which will use the before mentioned service to return the boat object from the database.

Frontend

Now that an endpoint has been created to get the current boat, it is possible to show it in the frontend.

To get the data from the backend, a service is created. This service will contain a variable for the boat, as well as a method for getting the boat data from the backend. This method can then be called continuously, to update the data as the boat moves.

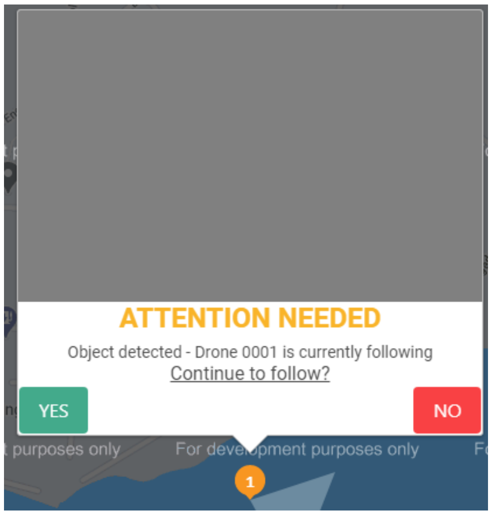
A.2.4 Notification Box

This section describes the implementation of the user story: *As a user I would like to be notified when a UAV has found an object in the water*.

The notification box opens when an object is found and shows the video feed and asks the user to keep following or not. If the user presses yes the drone will keep following the object, and if the user presses no it will continue on its path. The notification box can be seen on Figure A.2.

Backend

A backend endpoint was made to receive the input from the frontend and updates the drone with to the input received from the user. The endpoint receives a drone id and a boolean, true for keep following and false for continue to search its path.



Object detected - Drone 0001 is currently following
Continue to follow?

Figure A.2: Drone notification box

Frontend

A new component were implemented for the new info box. An info box were already implemented for the missing requirements, explained in Section 4.2.4. The new notification uses the same component, but with a different design. The notification box needed to have the video feed, a small description and two buttons. An if-statement were made in the loop of making the info boxes and if the drone status were *ActionNeeded*, the new notification box will pop up instead of the normal info display box.

```html
1 <!-- ACTION NEEDED INFO BOX -->
2   <agm-snazzy-info-window *ngIf="drone.status === 7" [maxHeight]="400"
       [maxWidth]="400" [isOpen]="true" [showCloseButton]="false">
3     <ng-template>
4       <app-action-needed-info-box [inputDrone]="drone"></app-action-needed-info-box>
5     </ng-template>
6   </agm-snazzy-info-window>
```

Listing A.11: Html - The if-statement for notification box

The if-statement can be seen on line 16 in Listing A.11, and are true if the *drone.status* is equals to 7, which is the enumerator corresponding to *ActionNeeded*. If it is true, the *agm-snazzy-info-window* component is used, defining the size and that the window is open and can not be closed in any other way than than responding using the buttons. The design of the info window is defined using the new component made called *action-needed-info-box*, which receives the drone.

A.2.5 Missing Person Prediction

This section describes the implementation of the user story: *As a user I would like to see the systems prediction on where the missing person is*.

The backend will provide a list of zones on the map, from a given start location, where it is predicted that the missing person will be at a given time. The frontend will then show these zones when the specified time has been reached.

Backend

To provide the predictions about where the missing person is, a data structure for this prediction is needed. For this a model was created, which can be seen in Listing A.12.

```
public class PersonPrediction
{
    public PersonPrediction(List<Coordinate> coords, int time)
    {
        predictedZone = coords;
        timeUntilPrediction = time;
    }

    public List<Coordinate> predictedZone { get; set; }
    public int timeUntilPrediction { get; set; }
}
```

Listing A.12: The *PersonPrediction* Model

The model for a prediction contains a list of coordinates, which is the polygon on the map where the person is predicted to be. It also contains an integer value for the time in seconds until this prediction should be shown on the map.

To get the prediction data a service was created. The service has one method, which is called *GetPrediction*. This method currently provides a static set of predictions, which can be used with the demo. It can be further extended in the further if a prediction system is created.

The service is then used in a controller, to create an endpoint for the frontend to access the data. The controller is given the endpoint *api/person*. The controller has one GET endpoint, which returns the list of static predictions from the service.

Frontend

To retrieve the predictions in the frontend, a service was created. This service has two variables for storing the predictions. One list for storing future predictions, and another for storing active predictions.

All of the predictions are then received from the before mentioned endpoint, and initially stored in the list for future predictions. To check if any of the predictions should be moved to the list of active predictions, a method is periodically called once per second. This method can be seen in Listing A.13.

```
private checkActivePredictions(): void {
  // Unsubscribe if there are no more predictions.
  if (this.missingPersonPrediction.length === 0) {
    this.subscription.unsubscribe();
  } else {
    this.missingPersonPrediction.forEach((prediction) => {
      let target = new Date(this.startTime.getTime() +
          (prediction.timeUntilPrediction) * 1000);
      let current = new Date();

      // Check if we have passed the time to show the prediction.
      if (current > target) {
        this.activePredictions.push(prediction);
        this.missingPersonPrediction.shift();
      }
    });
  }
}
```

Listing A.13: The *checkActivePredictions* method

The method checks each of the future predictions and if the target time has been reached, the prediction will be added to the active list.

To show the predictions on the map, an *agm-polygon* is added for each of the predictions in the active list. This can be seen in Listing A.14.

```
<ng-container *ngFor="let zone of missingPersonService.activePredictions">
  <agm-polygon
    [paths]="zone.predictedZone"
    [fillColor]="missingPersonZoneColor"
    [fillOpacity]="0.5"
    [strokeOpacity]="0.0"
  ></agm-polygon>
</ng-container>
```

Listing A.14: Adding the preditions to the map

A.2.6 Demo simulation

This section describes the implementation of the user story:*As a developer I should be able to create a set of instructions for a demo simulation*.
To conduct a user study on the interface during a SAR operation, a demo simulation would be beneficial. As the user study can be conducted in a controlled environment, with full control of

all the variables, and be able to comparing the results between the participants. The implementation of the simulation is in the backend, and is done through modifying the data of the UAVs in the database. As the frontend continuously listens on the data in the database through the backend, it will be viewed as it happens live in the frontend.

The first step in the backend was to create an endpoint to start the demo. For this a *DemoController* was created that consisted of a endpoint to start the demo. The endpoint is a GET request that returns a Ok response. When the endpoint is called it calls the *DemoService*, which consist of a *StartDemo* method. The *StartDemo* method is the main method, that starts the simulation.

The simulation depends on a predefined set of instructions, which is defined in a JSON file. For this first initial setup of the demo, it currently supports instructions for the UAVs. As seen in Listing A.15 in the JSON file on line 3, these instructions are defined for the UAV with the id 0001. The first instructions are defined on line 6-13, which are the positions that the UAV should between. Next on line 14 is the start time when the UAV should move in the simulation, such as if the UAV should start moving between these positions 30 seconds after the demo have started. A duration of the time it should take for the UAV to move between these positions is defined on line 15. The state during this duration is defined on line 16. The battery of the UAV that should be changed during the duration is defined on line 17-18. Lastly on line 19-20 is the altitude of the UAV that should be transitioned during the duration. These are all the variables that can be defined, and by creating a list of these it creates a simulation of the environment of the drones moving around during a SAR operation. As seen on line 22. there is space for a list of multiple of these instruction definitions. Line 25 is for multiple different of drones.

```json
1  [
2    {
3      "DroneID": "0001",
4      "Positions": [
5        {
6          "PositionA": {
7            "Latitude": 57.05267134366345,
8            "Longitude": 9.918667165444104
9          },
10         "PositionB": {
11           "Latitude": 57.05267134366345,
12           "Longitude": 9.918667165444103
13         },
14         "Start": "00:00:00",
15         "Duration": "00:00:10",
        "State": "Launching",
        "BatteryA": 100,
        "BatteryB": 96,
        "AltitudeA": 0,
        "AltitudeB": 70
      },
      ...
    ]
  },
  ...
]
```

Listing A.15: Drone simulation instructions

When the demo simulation starts the first step is loading all these instructions from the JSON file. This is done by reading the complete file and then deserializing it to a list of objects with the Newtonsoft JsonConvert. For this a *SimulationJson* model was created that consist of the UAV ID and a list of *SimulationPathJson*. The *SimulationPathJson* contains the previous mentioned definitions from the A.15.

```
public class SimulationJson
{
    public string DroneID { get; set; }
    public List<SimulationPathJson> Positions { get; set; }
}

public class SimulationPathJson
{
    public Position PositionA { get; set; }
    public Position PositionB { get; set; }
    public TimeSpan Start { get; set; }
    public TimeSpan Duration { get; set; }
    public DroneStatus State { get; set; }
    public int BatteryA { get; set; }
    public int BatteryB { get; set; }
    public int AltitudeA { get; set; }
    public int AltitudeB { get; set; }
}
```

Listing A.16: Drone Simulation model Json

After having loaded the JSON file into an object, a thread is created to continuously save any changes to the database each second. This is done to make all the UAVs update at the same time, and not having them update out of sync. Next the list of the instructions are iterated through as seen in Listing A.17 line 1. For each instruction, a drone id is defined, which is used to find the specific UAV that should be altered in the database, as seen on line 4. With the drone object, it is now possible to generate all the movements from the simulation positions, and this is seen on line 6-9. Each path is used to generate a simulation instruction with the *GenerateMove* method seen in Listing A.18. After all the instructions have been added to the list of *droneSimulation* on line 8 in Listing A.17, a thread is created to run the simulation and move the drones on line 11-15 using the *MoveDrone* method which can be seen in Listing A.19.

```
 1  foreach (SimulationJson simulation in data)
 2  {
 3      List<DroneSimulation> droneSimulation = new List<DroneSimulation>();
 4      Drone drone = await drones.FirstOrDefaultAsync(d => d.UUID ==
            simulation.DroneID);
 5
 6      foreach (SimulationPathJson path in simulation.Positions)
 7      {
 8          droneSimulation.Add(GenerateMove(drone, path));
 9      }
10
11      Thread thread = new Thread(delegate ()
12      {
13          MoveDrone(droneSimulation);
14      });
15      thread.Start();
16      }
17  }
18
19  return QueryResult.OK;
```

Listing A.17: Drone Simulation logic

The *GenerateMove* method is used to generate the movements between the positions, the smooth transition between the states, battery percentage, and altitude. By calculating the distance between the two positions given in the demo instructions, on line 8 in Listing A.18 and then dividing it with a speed on line 11 a number of positions for the transition can be generated. By iterating through the number of positions that should be generated for the transition on line 13-23, the current state for that number in the simulation can be generated. The position of the drone is calculated by taking the bearing and then adding a number of meters from the initial position A, line 17. The time of the state in the simulation is calculated on line 18. The state is simply the chosen instruction state, line 19. The battery percentage is calculated on line 20, and similarly the altitude of the drone is calculated on line 21.

```
private DroneSimulation GenerateMove(Drone drone, SimulationPathJson path)
{
    DroneSimulation droneSimulation = new DroneSimulation(drone);
    List<Simulation> simulationPath = new List<Simulation>();

    int speed = 10;

    double distance = GetDistance(path.PositionA, path.PositionB);
    double bearing = DegreeBearing(path.PositionA, path.PositionB);

    int number_of_positions = (int)Math.Ceiling(distance / speed);

    for (int i = 0; i < number_of_positions; i++)
    {
        Simulation simulation = new Simulation();
        double value = distance / number_of_positions * i / 1000;
        simulation.Position = GetPositionFromDistanceAndBearing(path.PositionA,
            value, bearing);
        simulation.Time = path.Start + i * (path.Duration / number_of_positions);
        simulation.State = path.State;
        simulation.Battery = path.BatteryA - (i * (path.BatteryA - path.BatteryB) /
            number_of_positions);
        simulation.Altitude = path.AltitudeA - (i * (path.AltitudeA -
            path.AltitudeB) / number_of_positions);
        simulationPath.Add(simulation);
    }

    droneSimulation.Simulation = simulationPath;

    return droneSimulation;
}
```

Listing A.18: GenerateMove Simulation logic

To then move the UAV and make the simulation active, the *MoveDrone* method is used. As stated earlier the UAV object is retrieved from the database and is continuously saved every second, which means that by modifying the values of the referenced UAV it will update the UAV in the database, and then result in the frontend retrieving the updated UAV object. By first starting a stopwatch, to ensure that the simulation runs within the expected times. Then, through iterating the generated UAV simulation a while loop waits for the elapsed time to have passed the expected time for the current state of the UAV in the simulation. When the expected time have passed on line 10-34, the UAV state is updated on line 14-18.

```
private void MoveDrone(List<DroneSimulation> droneSimulation)
{
    Stopwatch stopWatch = new Stopwatch();
    stopWatch.Start();

    foreach (DroneSimulation droneSimulationPath in droneSimulation)
    {
        foreach (Simulation sim in droneSimulationPath.Simulation)
        {
            while (sim.Time.TotalSeconds > stopWatch.ElapsedMilliseconds / 1000) //
                ensures that the demo does not run faster than expected
            {
                Thread.Sleep(200);
            }
            droneSimulationPath.Drone.CurrentPosition = sim.Position;
            droneSimulationPath.Drone.CurrentPosition.Altitude = sim.Altitude;
            droneSimulationPath.Drone.BatteryPercentage = sim.Battery;
            droneSimulationPath.Drone.LastUpdate = DateTime.Now;
            droneSimulationPath.Drone.Status = sim.State;
        }
    }

}
```

Listing A.19: MoveDrone Simulation logic

A.2.7 Sprint Review

At the end of the second sprint, the traditional sprint review meeting were held. It was concluded in the meeting that all user stories planned in the sprint planning meeting was completed successfully.

User story	Story points	Completed
As a user I would like to see the FOV of the UAVs on the map	2	✓
As a user I would like to see the rescue boat on the map	1	✓
As a user I would like to see the systems prediction on where the missing person is	3	✓
As a user I would like to be notified when a UAV has found an object in the water	2	✓
As a developer I should be able to create a set of instructions for a demo simulation	5	✓

Table A.4: Sprint 2: Completed User Stories

A.2.8 Sprint Retrospective

In Table A.4 it is seen that all the selected user stories in Sprint 2 were completed, giving a total of 13 user story points completed during Sprint 2. However due to more time now can be spent on developing in the next sprint. It is expected that more than 13 story points can be completed during the sprint period. In Section A.1 for Sprint 1, it is mentioned that more than 13 story points could be completed for next sprint. However, during Sprint 2 the related works chapter was also being researched and written, hence the 13 story points for Sprint 2 as well.

A.3 Sprint 3

This section contains a log of the third sprint which took place from March 17th to March 26th. The sprint includes implementation of the more demo components, such as in the frontend showing mission information, and backend is able to simulate the boat

A.3.1 Sprint Planning

This section covers the sprint planning meeting held by the scrum team at the beginning of Sprint 3. The user stories selected for Sprint 3 is seen in Table A.5.

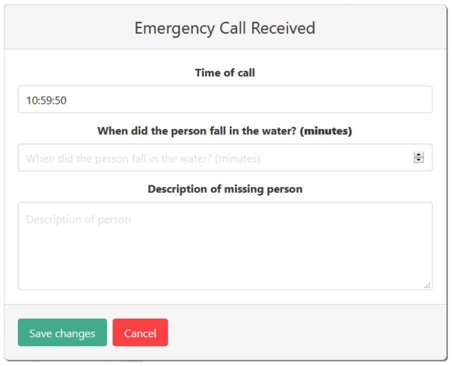
User story	Story points	System
As a developer I would like to be able to reset the demo from the frontend	2	Backend
As a developer I would like to have the JSON data for the movement and states of all the UAVs during the demo	1	Backend
As a developer I would like to have the JSON data for the movement and states of the boat during the demo	½	Backend
As a developer I would like to have the JSON data for the zone prediction of the missing person during the demo	1	Backend
As a developer I want to be able to simulate the movement of the boat to the found person	2	Backend
As a user I would like to see the boat moving to the found person	1	Frontend
As a user I would like to know when the rescue boat will arrive at the location of the missing person	2	Frontend
As a user I would like to have a form to input information from the distress call	2	Frontend
As a user I would like to have a box with mission critical information	2	Frontend
As a user I would like to have a box with weather information	1	Frontend
As a developer I would like to have a video of a found person and bird in the water	2	Frontend

Table A.5: Sprint 3: Selected User Stories

The total amount of story points needed to implement all user stories is 16.5, which the scrum team is confident can be achieved during this sprint.

A.3.2 Distress Call Form

This section describes the implementation of the user story *As a user I would like to have a form to input information from the distress call*.


Emergency Call Received

Time of call

10:59:50

When did the person fall in the water? (minutes)

When did the person fall in the water? (minutes)

Description of missing person

Description of person

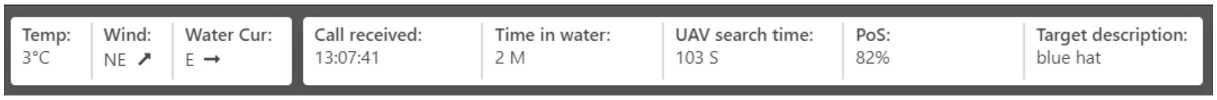
Figure A.3: Distress Call Modal

To enable the user to log any relevant information when an emergency call is received, a form was needed. For this a new component was created called *Receive Call Component*. This component contains a Bulma modal, which will be enabled when the user clicks the 'Setup' button in the header. The modal contains a form with three inputs: the time of the call, how long the missing person has been in the water, and a short description of the person. The modal can be seen in Figure A.3.

When the user clicks on the 'Save changes' button, the values of the input fields are saved in corresponding variables in the *Mission Info Service*.

A.3.3 Mission Box with Critical Information and Weather Information

This section describes the implementation of the user story *As a user I would like to have a box with mission critical information* and *As a user I would like to have a box with weather information*. The info is currently displayed when the call is received, how long the person has been in the water, how long the UAVs have been searching, the PoS (probability of success) and the description of the target. This information is filled out and calculated after the call received form is filled out by the operator, as described in the previous Section A.3.2.



Temp:	Wind:	Water Cur:	Call received:	Time in water:	UAV search time:	PoS:	Target description:
3°C	NE ↗	E →	13:07:41	2 M	103 S	82%	blue hat

Figure A.4: Mission Info Box

The weather information box displays the current temperature, wind direction, and the water currents. Both the mission info and the weather box are placed in the header of the dashboard and is placed at the top of the screen.

Two new components were created for each box and were added in the header file HTML.

The components are displaying the information from the *missionService*, and as of right now are the weather information static. The call received time is logged and displayed, the time in water is received from the operators input. The UAV search time is dependent on how long the UAVs take to search an area. The PoS is a calculation dependent on how long the person has been in the water and the current weather situation. The target description is also an input from the operator.

A.3.4 Reset Demo

In this section the implementation of the user story *As a developer I would like to be able to reset the demo from the frontend* is described. This feature was needed, so it would be possible to quickly reset the demo from the frontend, both during development but also when the demo has to be presented to the test participants.

To implement this, some changes were made to the existing *'Recall Search'* endpoint. When the existing logic for the endpoint has been run, an additional function is added. This function is called *SetInitialPositions*, and it is created in the *DemoService*. The function retrieves all UAVs from the database, and sets their positions to a set of predetermined positions along the shoreline of Aalborg Harbour, as well as setting all the UAVs statuses to charging.

A.3.5 Simulation JSON

This section will describe the three user stories that creates the instructions for the simulation. To help create the positions in the JSON file Google Maps were used. Google Maps allows to create a set of positions that can be saved and shared, which made it easy to discuss and move the positions around to create a set of waypoints for the UAVs and the boat. To create the positions of the zones for the prediction zone area, the frontend were used to draw zones and then log the positions that the area consisted of. This made it easy to draw tons of zones for the user story described in Appendix A.3.5.

Simulation UAV Movement JSON

The user story *As a developer I would like to have the JSON data for the movement and states of all the UAVs during the demo* implements a JSON file with all the states of each UAV that has been made in this sprint. The structure of each state can be seen in Listing A.15. Each UAV have a set of instructions to follow during a demo and the instructions tell the UAV when to be launching, searching, and when an action is needed.

Simulation Boat Movement JSON

The user story *As a developer I would like to have the JSON data for the movement and states of the boat during the demo*, also implements a JSON file with a similar structure as the UAVs. It contains a set of instructions, with an estimated time of arrival for the boat when the missing person have been found and are used to make it search a realistic area until it receives the position of the missing person in the water.

Simulation Prediction Zone JSON

The user story *As a developer I would like to have the JSON data for the zone prediction of the missing person during the demo* implements the prediction zone JSON file which contains all the positions for each area. Each area is an array of positions and the file contains and array of all the areas.

A.3.6 Simulation of Boat Movement

In this section the implementation of the simulation of boat movement in the demo is discussed for both the frontend and backend. This includes three user stories, for the backend *As a developer I want to be able to simulate the movement of the boat to the found person* and the frontend *As a user I would like to see the boat moving to the found person* and *As a user I would like to know when the rescue boat will arrive at the location of the missing person*.

Backend

The implementation of simulating the boat movement is similar to the implementation of the simulation of the UAV movement described in Appendix A.2.6. By additionally loading the simulation boat JSON instructions that is discussed in Appendix A.3.5. To load the JSON instructions, models were created to deserialize the JSON into a object, and this can be seen in Listing A.20 on line 14-27. The boat instructions is similar to the UAV instructions, and primarily differentiates with having less variables and an estimated time variable.

```
1  public class BoatSimulation
2  {
3      public Boat Boat { get; set; }
4      public List<SimulationBoat> Simulation { get; set; }
5  }
6
7  public class SimulationBoat
8  {
9      public Position Position { get; set; }
```

```
    public TimeSpan Time { get; set; }
    public TimeSpan EstimatedTime { get; set; }
}

public class SimulationBoatJson
{
    public string BoatID { get; set; }
    public List<SimulationBoatPathJson> Positions { get; set; }
    public TimeSpan EstimatedTime { get; set; }
}

public class SimulationBoatPathJson
{
    public Position PositionA { get; set; }
    public Position PositionB { get; set; }
    public TimeSpan Start { get; set; }
    public TimeSpan Duration { get; set; }
}
```

Listing A.20: Boat simulation model JSON

In Listing A.21 the logic of computing and moving the boat in the simulation is seen. It starts off by getting a reference point to the boat in the database on line 2-3. From that using the loaded JSON instructions it is possible to compute the boat simulations. On line 7 each of the instructions is computed into the boat simulation, and this is achieved by using a method similar to calculating the UAV movement as in Listing A.18. It is adjusted to work with the different boat variables, such as the estimated time. With the boat simulations, a thread is started that moves the boat around as instructed. Additionally, when a person is found, a new set of instructions is calculated from the current position of the boat to the found persons position, to start moving the boat towards that location in the simulation, as seen on line 14-25.

```
1  List<BoatSimulation> boatSimulation = new List<BoatSimulation>();
2  DbSet<Boat> boats = dbContext.Boats;
3  Boat boat = await boats.FirstOrDefaultAsync();
4
5  foreach (SimulationBoatPathJson path in dataBoat.Positions)
6  {
7      boatSimulation.Add(calculateBoat(boat, path, new TimeSpan(0,0,0)));
8  }
9
10 Thread threadBoat = new Thread(delegate ()
11 {
12     moveBoat(boatSimulation);
13
```

```
    while (_personPosition != null)
    {
        // Move boat towards person.
        boatSimulation = new List<BoatSimulation>();
        SimulationBoatPathJson path = new SimulationBoatPathJson();
        path.PositionA = new Position(boat.Latitude, boat.Longitude, 0,
            boat.Heading);
        path.PositionB = _personPosition;
        path.Start = new TimeSpan(0, 0, 5);
        path.Duration = new TimeSpan(0, 0, 30);
        boatSimulation.Add(calculateBoat(boat, path, dataBoat.EstimatedTime));
        moveBoat(boatSimulation);
        boat.EstimatedTime = new TimeSpan(0, 0, 0);
    }
});
```

Listing A.21: Boat simulationlogic

Frontend

With the backend simulating the movement and the frontend continuously receiving updates from the backend, it needs to also visualize the direction that the boat is heading and the estimated time of arrival to that location.

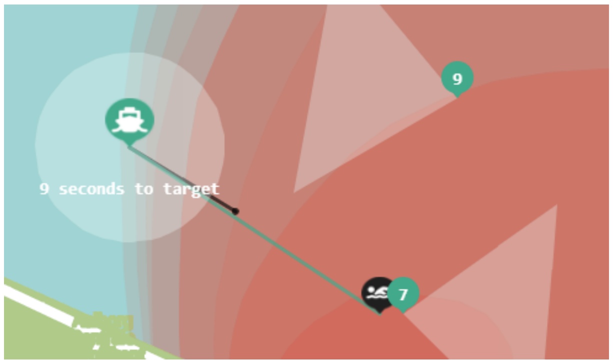


Figure A.5: Boat direction and estimated time of arrival

The first user story *As a user I would like to see the boat moving to the found person* is implemented by having a line showing from the boat to the last known location of the missing person. This can be seen on Figure A.5 with the green line between the boat and the missing person icon. The implementation is shown in Listing A.22 and is done by creating the polyline between the missing person location and location of the boat.

```
<agm-polyline *ngIf="boatService.travelToPerson && boatService.boat != null &&
    boatService.boat.estimatedTime.totalSeconds !== 0"
      [strokeColor]="boatService.boatColor"
      [strokeOpacity]="0.7"
      [zIndex]=9999>
      <agm-polyline-point [latitude]="boatService.boat.latitude"
          [longitude]="boatService.boat.longitude"></agm-polyline-point>
      <agm-polyline-point [latitude]="missingPersonService.personLocation.lat"
          [longitude]="missingPersonService.personLocation.lng"></agm-polyline-point>
</agm-polyline>
```

Listing A.22: Boat line to missing person

The second user story *As a user I would like to know when the rescue boat will arrive at the location of the missing person* is implemented by having the estimated time below the boat icon. This can be seen on Figure A.5 with the *9 seconds to target* text below the boat icon. The implementation is shown in Listing A.23 and is achieved by creating a marker below the boat location with the label of the total seconds of arrival received from the backend.

```
<agm-marker *ngIf="boatService.travelToPerson && boatService.boat != null &&
    boatService.boat.estimatedTime.totalSeconds !== 0"
      [latitude]="boatService.boat.latitude - 0.0002"
      [longitude]="boatService.boat.longitude"
      [label]="{color: 'white', text: boatService.boat.estimatedTime.totalSeconds +
          ' seconds to target', fontWeight: 'bold', fontFamily: 'monospace'}"
      [iconUrl]="{url: boatIcon, scaledSize:{height:0,width:0}}"
    >
</agm-marker>
```

Listing A.23: Boat estimated time of arrival to missing person

A.3.7 Videos of Detected Objects

This section describes the implementation of the user story *As a developer I would like to have a video of a found person and bird in the water*.

To make the demo more realistic for the participants, some videos of objects in the water was needed when an UAV detected an object. It was determined that two videos were needed, one for when a person had been found, and one were an unwanted object had been found.

Creating the Videos

It was not possible to find two suitable videos, so they were created instead. This was done with Adobe Photoshop and Adobe After Effects.

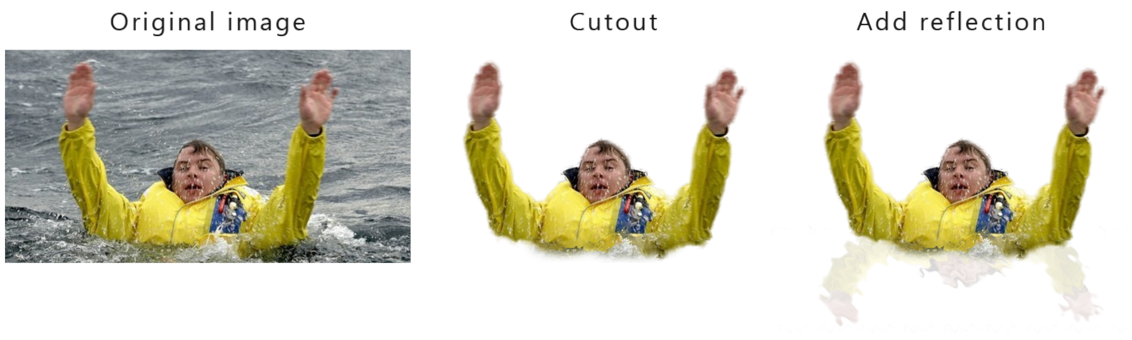


Figure A.6: Creating a object image for the video

Two objects with transparent backgrounds were made in Photoshop. A picture of a seagull was chosen for the unwanted object, and a suitable picture of a person in need was found as well. The backgrounds from these pictures were then cut, and a mirrored version of the cutout was placed below the image, to simulate a ripple effect in the water. An example of this process can be seen in Figure A.6.

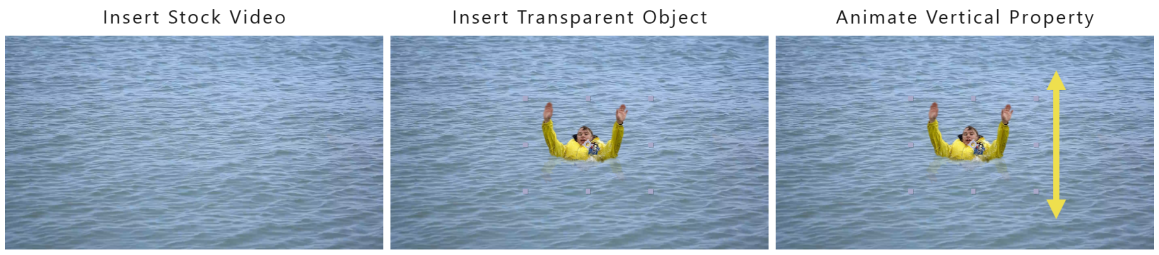


Figure A.7: Creating the final video

A stock video of an ocean was found, with a field of view that could pass as being from a stationary hovering UAV. The transparent objects were then placed above the video in After Effects, and the vertical position of these objects where animated to simulate the objects following the waves of the ocean. An example of this process can be seen in Figure A.7.

Showing The Videos

To show the videos during the demo, they had to be added to the *Action Needed Component*. A counter was added to the *Drone Service* to decide which image to show in the *Action Needed Component*. If the counter is 0, the video of the seagull will be shown, and if the counter is 1 the video of the person will be shown.

Currently, the counter is initially set to 0, so the first item will always be an unwanted object.

When the user discards this object, the counter is incremented with one, and the next object found will be the missing person.

A.3.8 Sprint Review

At the end of the third sprint, the traditional sprint review meeting were held. It was concluded in the meeting that all user stories planned in the sprint planning meeting was completed successfully.

User story	**Story points**	**Completed**
As a developer I would like to be able to reset the demo from the frontend	2	✓
As a developer I would like to have the JSON data for the movement and states of all the UAVs during the demo	1	✓
As a developer I would like to have the JSON data for the movement and states of the boat during the demo	½	✓
As a developer I would like to have the JSON data for the zone prediction of the missing person during the demo	1	✓
As a developer I want to be able to simulate the movement of the boat to the found person	2	✓
As a user I would like to see the boat moving to the found person	1	✓
As a user I would like to know when the rescue boat will arrive at the location of the missing person	2	✓
As a user I would like to have a form to input information from the distress call	2	✓
As a user I would like to have a box with mission critical information	2	✓
As a user I would like to have a box with weather information	1	✓
As a developer I would like to have a video of a found person and bird in the water	2	✓

Table A.6: Sprint 3: Completed User Stories

A.3.9 Sprint Retrospective

In Table A.6 it is seen that all the selected user stories in Sprint 3 were completed, giving a total of 16.5 user story points completed during Sprint 3.
The Scrum Team believes a similar amount of story points can be achieved during the next sprint.

A.4 Sprint 4

This section contains a log of the third sprint which took place from April 8th to April 16th. The sprint includes implementation of the more demo components and finalizing the demo for the user studies.

A.4.1 Sprint Planning

This section covers the sprint planning meeting held by the scrum team at the beginning of Sprint 4. The user stories selected for Sprint 4 is seen in Table A.7.

User story	Story points	System
As a developer I would like support for a folder for each demo scenario	2	Backend
As a developer I would like to have two more demo scenarios and a tutorial scenario	3	Backend
As a developer I should have an endpoint for the mission information to the frontend	2	Backend
As a developer I should receive the mission information from the backend	2	Frontend
As a user I would like to have a prediction of the missing person	1	Frontend
As a developer I should have data support for potential location of the missing person	1	Backend
As a developer I would like the backend to support endpoints for controlling the demos	2	Backend
As a developer I would like the frontend to have a demo panel for controlling the demos	3	Frontend
As a developer I would like to be able to pause the demo	½	Frontend / Backend
As a user I would like to be able to toggle the different UI elements	2	Frontend
As a user I would like to have a legend on the probability of the predicted zone	½	Frontend
As a user I would like to have a demo tutorial before using the system	2	Frontend

Table A.7: Sprint 4: Selected User Stories

The total amount of story points needed to implement all user stories is **21**, which the scrum team is confident can be achieved during this sprint.

A.4.2 Folder structure

This section describes the implementation of the user story *As a developer I would like support for a folder for each demo scenario*.

To support multiple scenarios in each demo the assets of the instructions of the demo were moved into a scenario folder. This change would make it possible to have an internal state in the backend of which scenario that would have to be loaded and used for the demo. By adding a endpoint that selects a scenario the internal state can be changed and and load the corresponding scenario.

```
using (StreamReader r = new
    StreamReader($"Assets/demos/scenario-{_selectedScenario}/simulation.json"))
{
    string json = r.ReadToEnd();
    return JsonConvert.DeserializeObject<List<SimulationDroneJson>>(json);
}
```

Listing A.24: Selecting a scenario

As seen in Listing A.24 the simulation asset is loaded with a dynamic selected *_selectedScenario* variable.

A.4.3 Dynamic Mission Information

This section describes the implementation of the user story *As a developer I should receive the mission information from the backend* and *As a developer I should have an endpoint for the mission information to the frontend*.

These user stories were needed to support multiple demo cases and in general just better practice than static variables in the frontend. A new endpoint were made in the backend fetching the information from a JSON file. Each demo case contains a JSON file which includes all the necessary information for that specific demo scenario. The data are then loaded in to the variables in the mission information service. All the places where the data were static was replaced with the corresponding variables.

The only place where the frontend needed some change to show the variables, were in the weather information box. Here the arrows change based on the current conditions.

```
<span class="icon fa-medium" [ngStyle]="{'transform':
    'rotate('+missionService.currentWaterDir+'deg)'}">
```

Listing A.25: Inline CSS with variables for rotating weather arrows

The arrow icon is by default showing zero degrees and will point north. If it needs to display south, it will be 180. If it needs to display west, it will be -45 and so on.

A.4.4 Toggle UI Elements

This section describes the implementation of the user story *As a user I would like to be able to toggle the different UI elements*.

The intended functionality was to have a panel from which the different UI elements could be toggled on and off by the operator. The appearance of the implemented panel can be seen in Figure A.8.

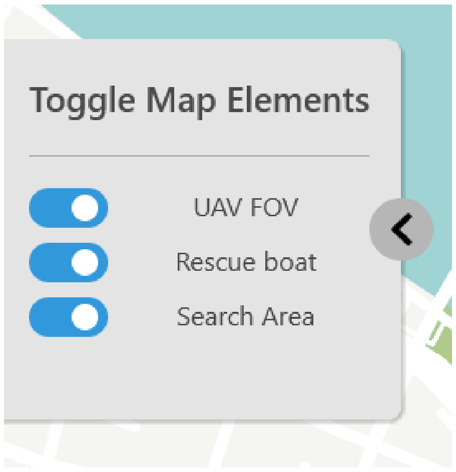


Figure A.8: Toggle UI Elements Panel

To implement the panel, a new component was created to display the panel and its inputs, as well as a service to keep track of what elements had been toggled. The component is composed of a panel, which is initially hidden, until the arrow button is pressed. It has three Bulma checkbox inputs, to control the UAV FOV, the rescue boat, and the search area. These three inputs are mapped to corresponding boolean variables in the UI service.

To allow the different UI elements to be toggleable, each of them is wrapped in a *ng-container*, which is set to be displayed whenever the corresponding boolean from the UI service is true. An example of how this looks can be seen in Listing A.26.

```
1 <!-- DRONE FOV -->
2 <ng-container *ngIf="uiService.showFOV">
3   <ng-container *ngFor="let drone of droneService.droneList">
4     <agm-polygon *ngIf="drone.status !== 1" [paths]="drone.ui.fovCone"
          [fillColor]="fovColor" [strokeOpacity]="0"
5       [zIndex]=9999></agm-polygon>
6   </ng-container>
7 </ng-container>
```

Listing A.26: Allowing the UAV FOV to be toggleable

The additional *ng-container*, is the one which can be seen on line 2. This container is made conditional with a **ngIf* statement, using the *showFOV* variable of the UI service. This container contains all the HTML elements that where there before it was made toggleable.

A.4.5 Heatmap Legend

This section describes the implementation of the user story *As a user I would like to have a legend on the probability of the predicted zone*.

The legend were added to increase readability of the heatmap. It is a small change that ensure that the user reads the heatmap correctly. The legend is displayed in the far right corner of the map and should only be shown when the heatmap is selected as the predictive element.

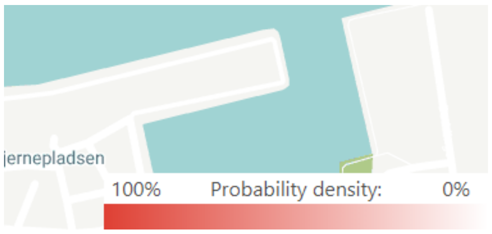


Figure A.9: Legend of heatmap

As seen on Figure A.9 it fades from darker red to white, showing that the darker the red, the higher probability. The legend is shown whenever the heatmap is chosen as the predictive element and this was easy to do, as the functionality for this were made in Section A.4.7.

A.4.6 Two More Scenarios and Tutorial Scenario

In this section the implementation of *As a developer I would like to have two more demo scenarios and a tutorial scenario* in the backend is discussed. The techniques previously used to get coordinates for the heatmap and person prediction were inefficient and creating the three more scenarios would require a lot of time. Using the MapService already imported in the project for drawing zones and everything else regarding the map, it was possible to also include a polyline part for drawing lines.

Getting Heatmap Coordinates Easier

Using the polygon drawings on the map, it was possible to extract the coordinates from the drawing. When a polygon drawing is saved, it logs the coordinates in the correct format to the console, reducing the workload of generating the zones significant. The saved zone is then displayed and it is possible to draw another zone upon it. This made the natural expanding of the heatmap zone easier to draw and copying the object from the console to the JSON file was straightforward.

Getting UAV Path and Person Prediction Coordinates Easier

The UAV paths on the first scenario were generated from Google Maps using their waypoint feature. This worked out fine, but were much easier using the polyline in our own interface. All the UAV paths were drawn and the coordinates were logged in the console in the correct format used in the demo JSON explained in Section A.3.5. This feature was also used to generate the person prediction coordinates by drawing the polyline of the person's path instead of the UAV's.

With the two new techniques, the implementation efficiency of creating new scenarios required way less time. This meant that creating the two new scenarios and the demo tutorial was straightforward with drawing realistic routes for the UAVs and zone predictions out from the defined scenarios. The two scenarios were defined to be a suicidal person that had been seen jumping from the *Limfjordsbroen* and a fisherman that has gone missing from his boat. The demo tutorial scenario was a smaller task of a single UAV moving around and finding both trash and a person, as well as showcasing the prediction zone and marker prediction.

A.4.7 Admin Panel

This section describes the implementation of the user story *As a developer I would like the frontend to have a demo panel for controlling the demos*.

The intended functionality was to have a panel which could be opened before a demo starts, to select a demo, and which predictive elements should be included in the demo. It should also be possible to reset the demo from this panel. The appearance of the admin panel can be seen in Figure A.10.

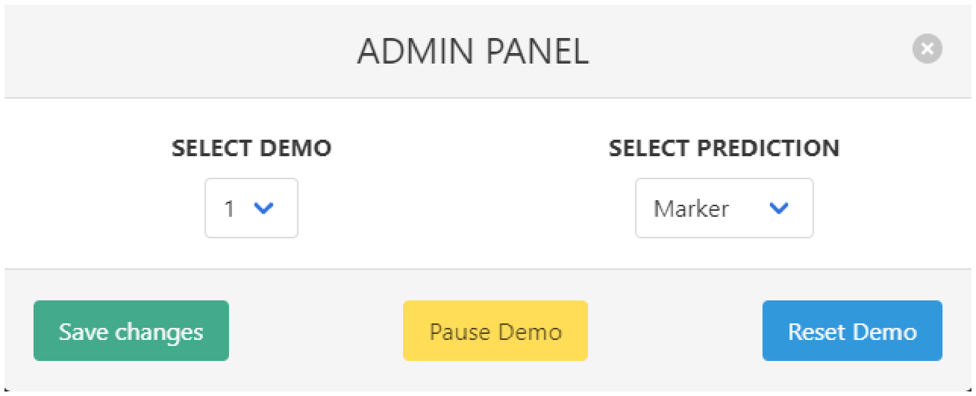


Figure A.10: The Admin Panel

To implement the panel, a new component was created to display the panel and the different inputs and buttons needed. To keep track of the users selections, the data was saved in the existing demo service.

The component contains a Bulma modal, which contains a form for selecting the different elements. The model is added to the dashboard, and is activated when the user presses the 'p' button. This was implemented by binding a function to the *document:keypress* event of the page, and checking if the correct key had been pressed.

The inputs from the form are bounded to variables in the demo service using *ngModel*, so the values can be sent to the backend later.

Further a pause button were added to support the user story *As a developer I would like to be able to pause the demo*. The implementation added the pause button seen on Figure A.10, and when the user presses the button, a request is sent to to the backend that pauses the demo.

To support these features in the admin panel in the frontend, the user user story *As a developer I would like the backend to support endpoints for controlling the demos* were necessary. The backend implementation required selecting a specific demo, being able to pause the demo and reset the demo. The first task of selecting a specific demo were implemented by having a endpoint that is able to change the internal state of selected demo that was discussed in Section A.4.2. To reset the demo a endpoint were created that calls the already existing reset method in the demo, that resets all states. Finally to pause the demo, it required further implementation in the demo, with a boolean *_demoPause* variable that holds the state of the demo is paused. By ensuring that all threads running will run continuously forever in a while loop and pause their stopwatches the demo can be paused. Lastly an endpoint were created to be accessible by the frontend to toggle pause the demo.

A.4.8 Potential Location

In this section the implementation of *As a developer I should have data support for potential location of the missing person* in the backend and *As a user I should be able to see the potential location of the missing person* in the frontend are discussed.

To support potential location of the missing person in the backend, a predicted person coordinate were added to the person prediction model and a confidence score. This change would would mean that the already existing person prediction for heat map which is retrieved in the frontend, will now also receive the potential location of the person. This then also requires changes to the *predictedArea* JSON file. For each zone defined in the *predictedArea* a corresponding potential location of the predicted person and a confidence score were set.

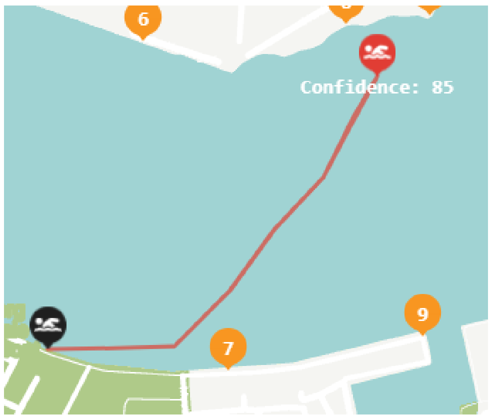


Figure A.11: Potential location of missing person

By making these changes to the model in the frontend, the data for potential location of the person is automatically retrieved the with zone predictions. This is then visualized in the frontend with showing a red marker on the map with a confidence score below. As well as showing the path from the last seen location of the person, linked with the other potential locations of the person to the current as seen on Figure A.11.

A.4.9 Sprint Review

At the end of the fourth sprint, the traditional sprint review meeting were held. It was concluded in the meeting that all user stories planned in the sprint planning meeting was completed successfully. Even though taking a considerable amount of story points more than last sprint, all the user stories were completed. This was the last sprint for this project and therefore it needed all the user stories completed to conduct the user study.

User story	**Story points**	**Completed**
As a developer I would like support for a folder for each demo scenario	2	✓
As a developer I would like to have two more demo scenarios and a tutorial scenario	3	✓
As a developer I should have an endpoint for the mission information to the frontend	2	✓
As a developer I should receive the mission information from the backend	2	✓
As a user I would like to have a prediction of the missing person	1	✓
As a developer I should have data support for potential location of the missing person	1	✓
As a developer I would like the backend to support endpoints for controlling the demos	2	✓
As a developer I would like the frontend to have a demo panel for controlling the demos	3	✓
As a developer I would like to be able to pause the demo	½	✓
As a user I would like to be able to toggle the different UI elements	2	✓
As a user I would like to have a legend on the probability of the predicted zone	½	✓
As a user I would like to have a demo tutorial before using the system	2	✓

Table A.8: Sprint 4: Completed User Stories

A.4.10 Sprint Retrospective

In Table A.8 it is seen that all the selected user stories in Sprint 4 were completed, giving a total of **21** user story points completed during Sprint 4. The increase of completed story points were achieved due to having more experience of our system, thus allowing us to complete the stories quicker without any issues.

A.5 Final Adjustments

This section will discuss the final adjustments that were found from the pilot study with our supervisor to be necessary to implement to conduct the user study. As it was originally planned that the user study would be conducted physically, but change of plans was made due to COVID-19, and meant that the user study had to be conducted virtually. This meant there were some necessary required changes and features to be implemented.

A.5.1 SSL Certificate

The first implementation was an SSL certificate. This was necessary to host the website online so that the participants could access the page without any issues. Both the website and backend had to be setup with an SSL certificate. Let's Encrypt Certbot was used to generate a certificate that could be used. With a SSL certificate a domain is also required, and this meant that the API URL in the frontend had to be updated. To support external communication the backend had to listen to any external IP requests, as well as port forwarding.

A.5.2 Hotkeys

Secondly, as the user study was being conducted on the participant's PC, and to control the user study hot key had to be implemented. As for controlling the user study through the interface would require larger changes, and with time being limited the hotkeys were a quick and easy fix to allow for a virtual study.

The selected demo scenario can be controlled externally by the examiner, but due to the technical implementation of the frontend, the selected demo is not automatically fetched when it is changed. This meant that a hotkey to fetch the newest data was created. When the user presses the *F* key on their keyboard, it fetches all the newest data. This feature would allow the examiner to ask the user to press the *F* key before beginning the user study, and ensure that they have the correct scenario data.

Additionally, hotkeys were made to switch between the different predictive elements. By pressing the *L* key the user would switch to the marker predictive element, the *H* key would switch to the heat-map predictive element and the *N* key would switch to no predictive element. The examiner would then be able to tell the participant to press the correct hotkey depending on which predictive element should be shown.

A.5.3 Pause Demo

During the demo the examiner should be able to pause the demo, to give the participants the SAGOT questionnaire. During the pause, the participant should not be able to see anything on the screen. The implementation of this was made by creating an endpoint in the backend that could be used in the frontend for consciously checking if the demo has been paused. When the examiner pauses the demo on his computer, the participants demo would freeze up, and show nothing as seen on Figure 6.1.

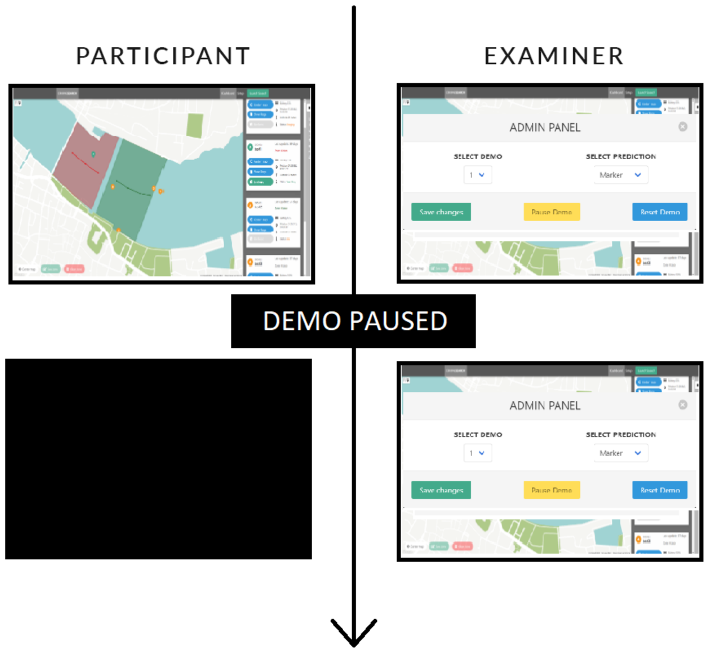


Figure A.12: Examiner pausing the demo

A.5.4 Saving Drawing Zones

Lastly, to know the different zones that are drawn during the user studies they would have to be sent to the backend and saved. The implementation to support this required a backend endpoint for receiving the zones. Further, the implementation was made to have a folder with the data from the user studies. In the data folder, a separate folder would be generated for each participant with the IP address as the unique identifier back to the participant. In this folder, a file for each scenario would be generated when they save their zone, with all the coordinates that the zone consists of. This folder structure can be seen in Listing A.27.

```
- Assets/
  - data/
    - 192.168.0.1/
      - 0.json
      - 1.json
      - 2.json
    - 192.168.0.2/
      - 0.json
      - 1.json
      - 2.json
```

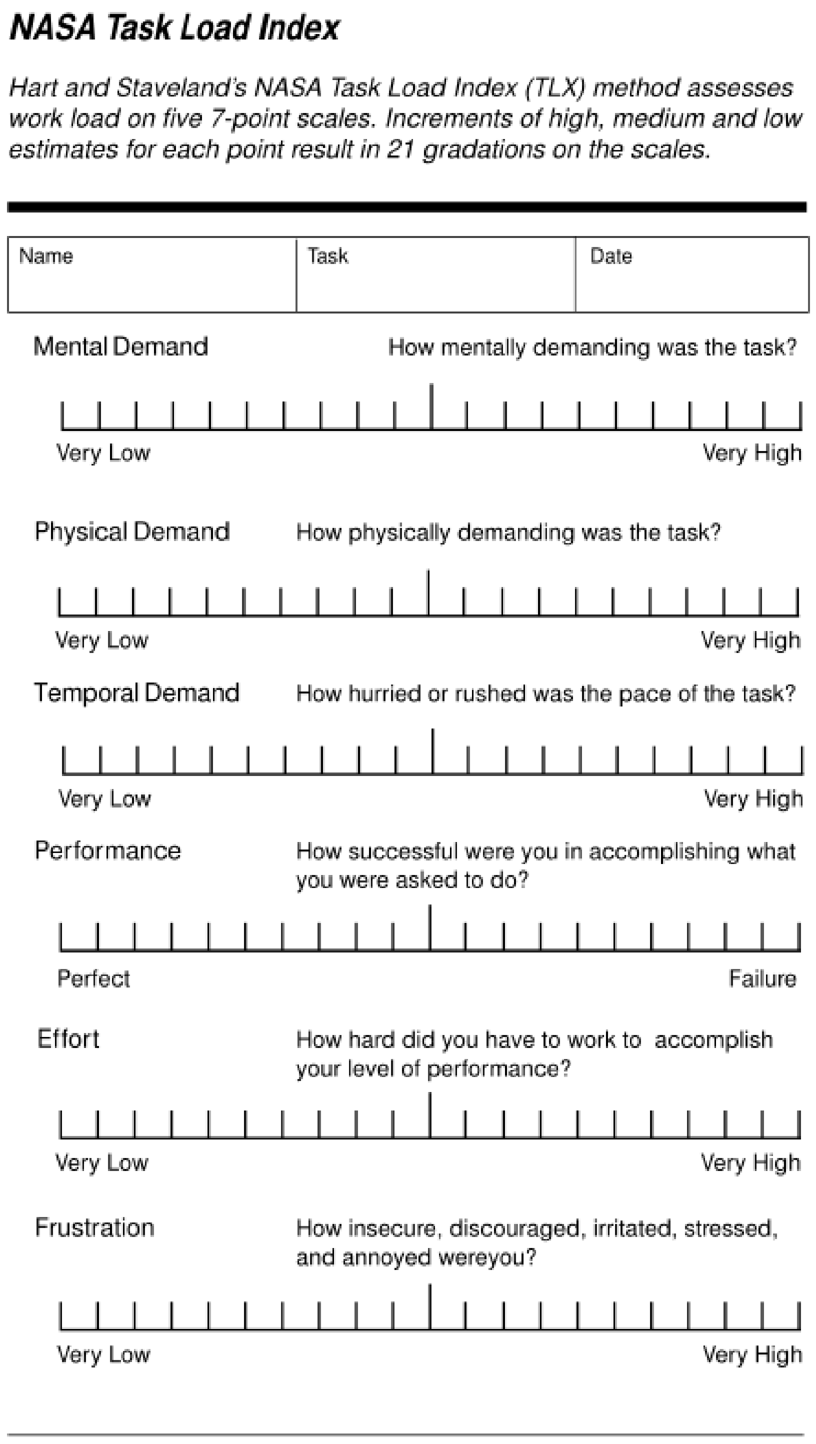
Listing A.27: Folder structure for saving data for each particpants

In the frontend this was made as to when the user saves their zone, it sends the post request with the zone data to the backend.

A.6 General Requirements

1. The system shall help the operator maintain a low mental workload.
2. The system shall help the operator maintain a high situational awareness.
3. The system shall have high reliability as it is important that it does not freeze or crash during a critical operation.
4. The system shall have high performance as time is critical.
5. The system shall provide a GUI capable of defining and adjusting search zones.
6. The system shall provide a GUI showing the position of each UAV on a map.
7. The system shall provide a GUI showing the status of each UAV in the system.
8. The system shall provide pop-ups during status changes to increase SA.
9. The system shall provide a REST API to allow the frontend to interact with the system.
10. The system shall provide the functionality to launch and recall the UAVs.
11. The system shall provide communication between the UAVs and the backend to provide video stream, GPS coordinates, and current status.
12. The interface shall not rely on scrolling, as it can make it difficult for the operator to find the relevant information.
13. The interface shall only present information in numbers if absolutely necessary.
14. The interface shall avoid showing too much information, as it decreases SA.
15. The interface shall show a precise alternative for GPS coordinates, as they are hard to communicate.
16. The interface shall show points of interest on the map to increase the SA.
17. The interface shall display UAV information next to the UAV's icon, such that operators do not need to search for the information.
18. The interface shall use instant pop-ups instead of notifications.

A.7 NASA Task Load Index - Scale



NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

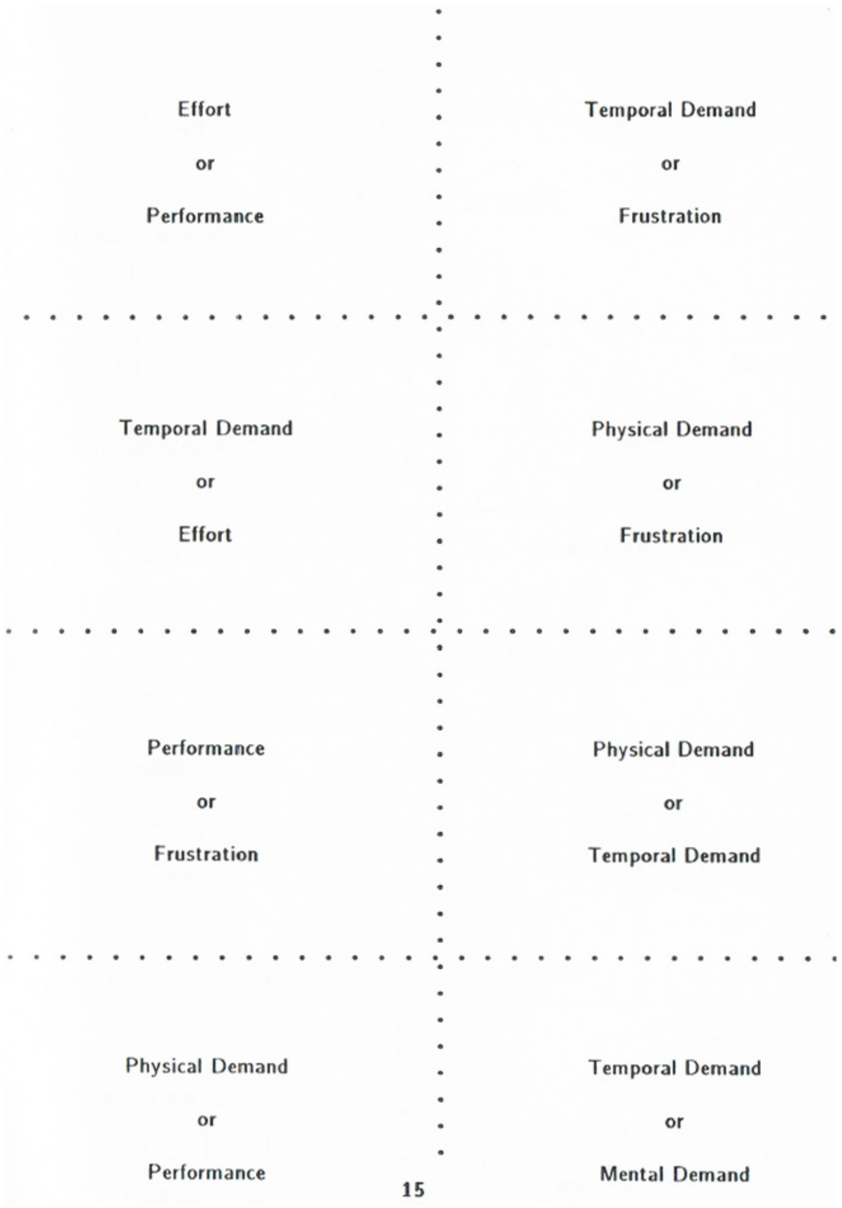
Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?

Very Low — Very High

Figure A.13: NASA Task Load Index (Gore, 2020)

A.8 NASA Task Load Index - Weighting



Effort

or

Performance

Temporal Demand

or

Frustration

Temporal Demand

or

Effort

Physical Demand

or

Frustration

Performance

or

Frustration

Physical Demand

or

Temporal Demand

Physical Demand

or

Performance

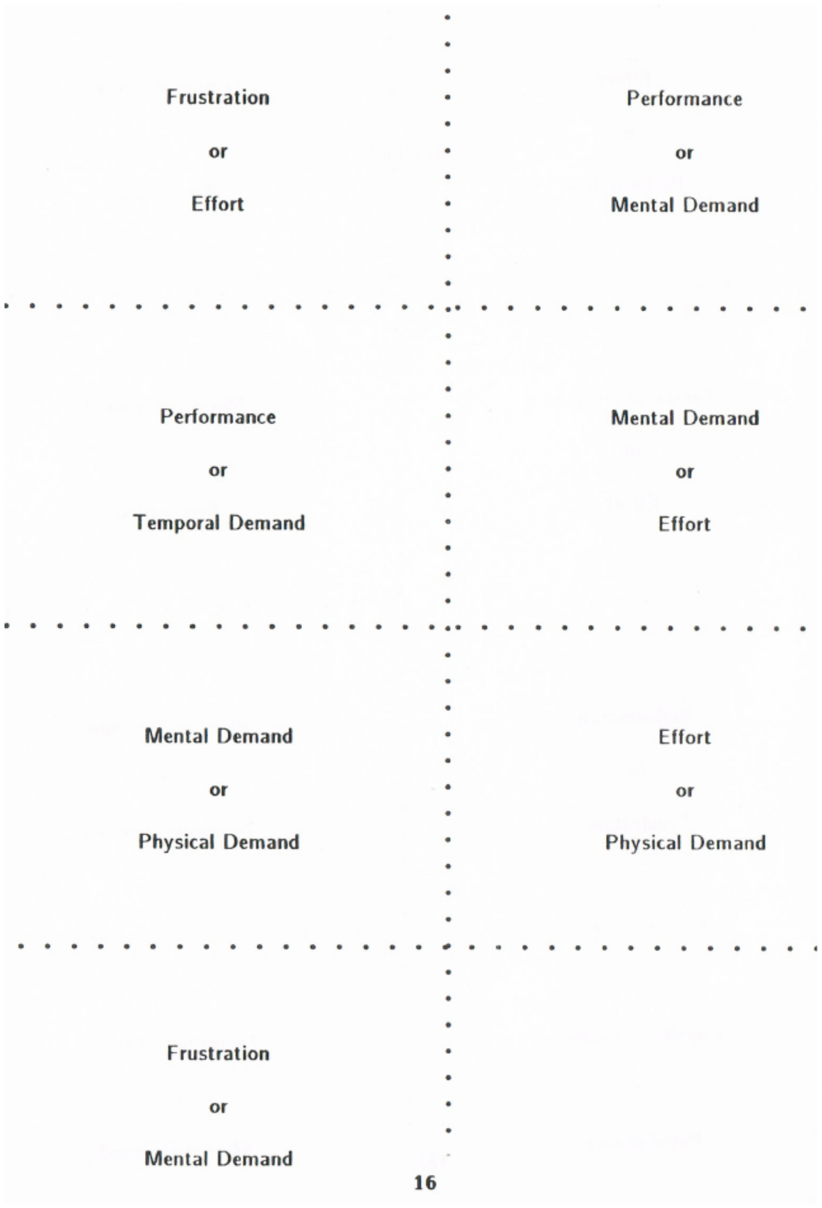
Temporal Demand

or

Mental Demand

15

Figure A.14: NASA TLX Weights (1 of 2) (NASA)



Frustration

or

Effort

Performance

or

Mental Demand

Performance

or

Temporal Demand

Mental Demand

or

Effort

Mental Demand

or

Physical Demand

Effort

or

Physical Demand

Frustration

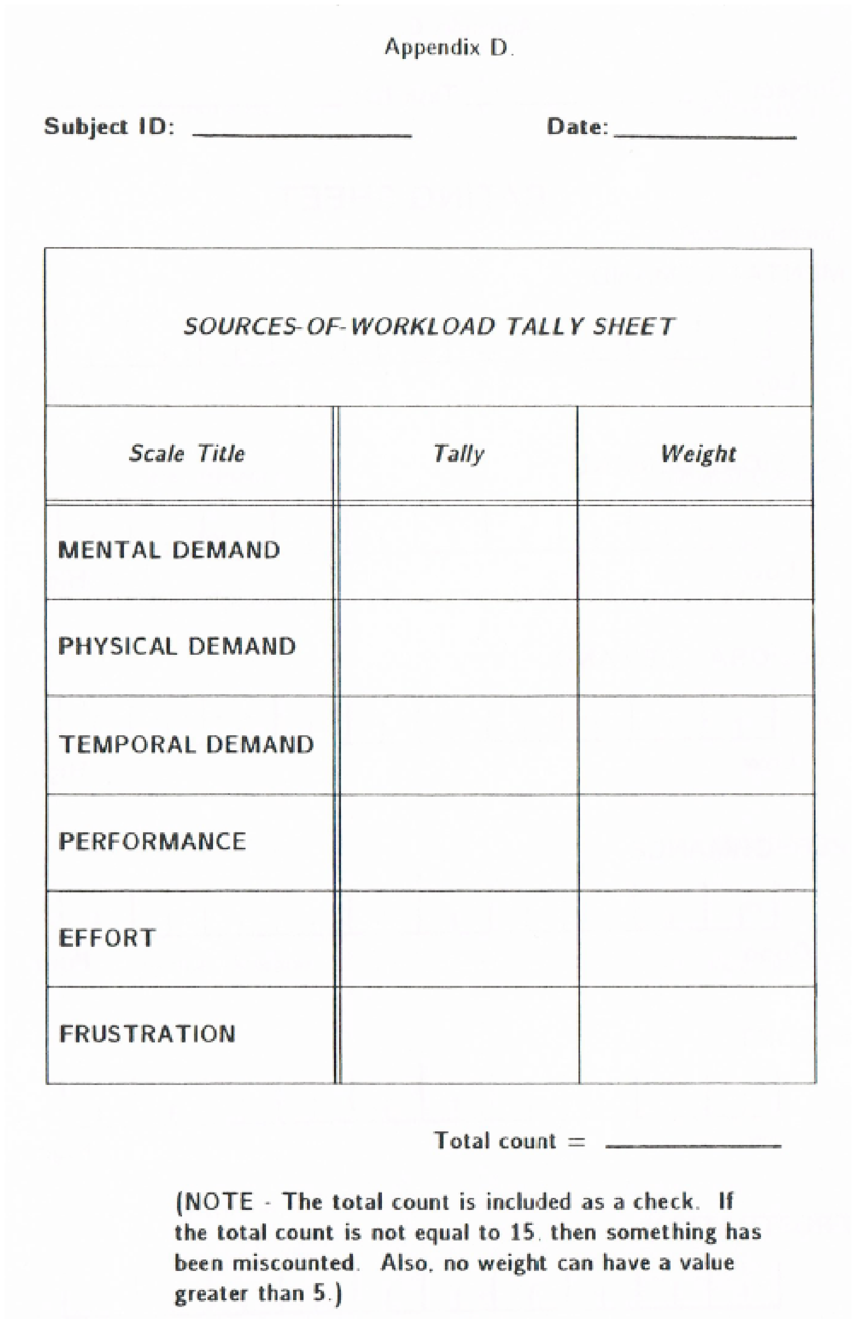
or

Mental Demand

16

Figure A.15: NASA TLX Weights (2 of 2) (NASA)

A.9 NASA Task Load Index - Tally Sheet



Appendix D.

Subject ID: ______________ **Date:** ______________

SOURCES-OF-WORKLOAD TALLY SHEET

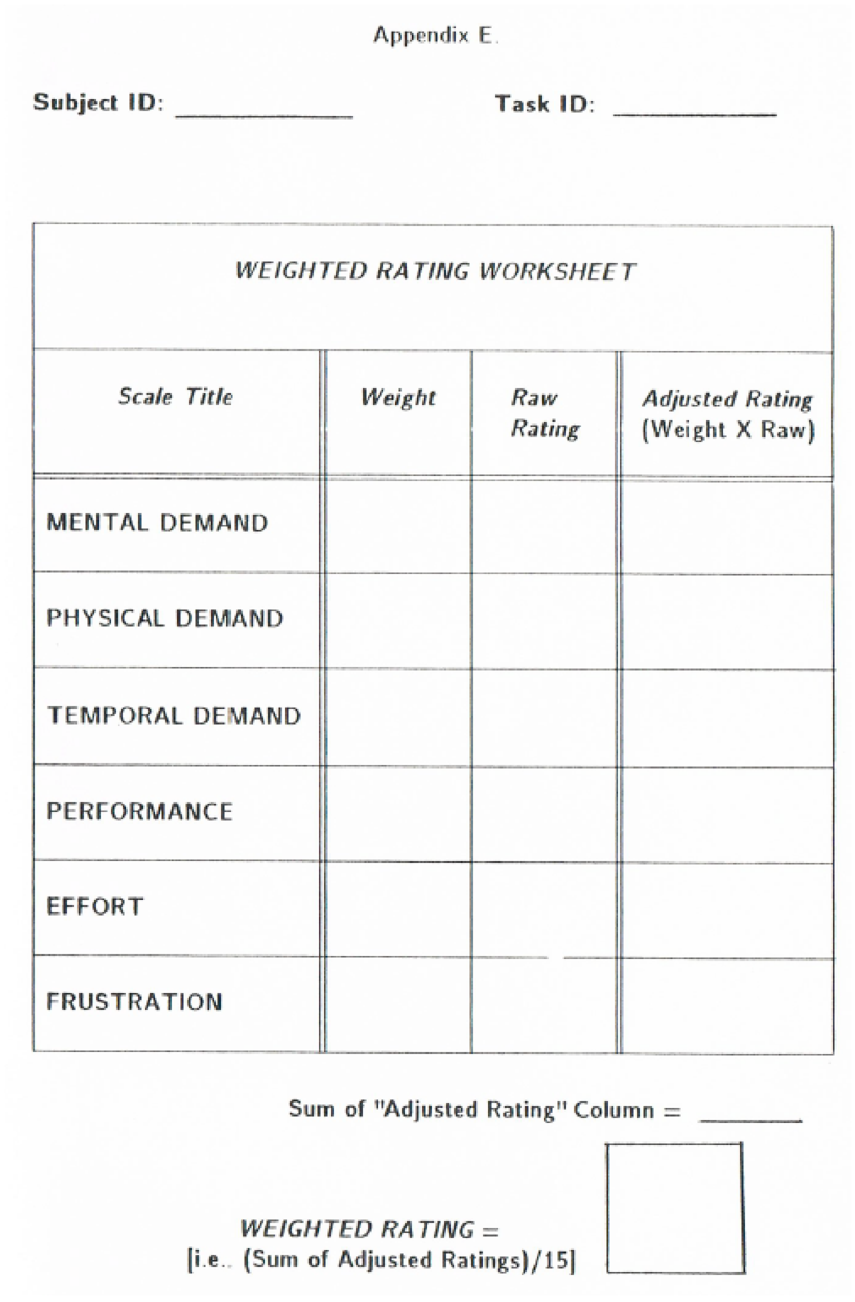
Scale Title	*Tally*	*Weight*
MENTAL DEMAND		
PHYSICAL DEMAND		
TEMPORAL DEMAND		
PERFORMANCE		
EFFORT		
FRUSTRATION		

Total count = ______________

(NOTE - The total count is included as a check. If the total count is not equal to 15, then something has been miscounted. Also, no weight can have a value greater than 5.)

Figure A.16: Sources-of-workload Tally Sheet (NASA)

A.10 NASA Task Load Index - Worksheet



Appendix E.

Subject ID: ______________ **Task ID:** ____________

WEIGHTED RATING WORKSHEET			
Scale Title	*Weight*	*Raw Rating*	*Adjusted Rating* (Weight X Raw)
MENTAL DEMAND			
PHYSICAL DEMAND			
TEMPORAL DEMAND			
PERFORMANCE			
EFFORT			
FRUSTRATION			

Sum of "Adjusted Rating" Column = ________

WEIGHTED RATING =
[i.e.. (Sum of Adjusted Ratings)/15]

Figure A.17: Weighted Rating Worksheet (NASA)

A.11 NASAS TLX Introductions

A.11.1 NASA TLX Rating Sheet Introduction

We are not only interested in assessing how well you perform during these scenarios but also how you experience the different scenarios. I will now explain the technique we will use to examine your experiences.

More specifically, we are examining the workload you experienced during each scenario. Workload is a difficult concept to define precisely but more simple to understand generally.

The factors that influence your experience of workload may come from the tasks themselves, your feelings about your performance, how much effort you put in, or the stress and frustration you felt.

We will measure these different factors by giving you a NASA TLX sheet after each ended scenario, where you can fill out how low or high each of the six factors was during the scenario.

On the form page that you have been provided, a definition of the six different factors can be found. Please take your time to read them carefully, and be sure to ask any questions if you are in doubt. If not, you may begin when you are ready.

A.11.2 NSA TLX Weighting Sheet Introduction

Now that you have filled out the rating sheet, we will need you to fill out the weighting sheet. One of the problems with assessing your experiences is that scales of this sort suffer from the tendency that people interpret them in different ways. For example, some people feel that mental or temporal demands are the essential aspects of workload, regardless of the effort they expended on a given task or the level of performance they achieved. Others may feel that if they performed well, the workload must have been low, and if they performed poorly, it must have been high.

The evaluation you are about to perform is a technique we can use to assess the relative importance of the six factors in determining how much workload you experienced. In the procedure, you will be presented with a series of pairs of rating scales. (For example, Effort vs. Mental Demands) For each pair, you will need to select which rating scale is the essential contributor to workload for the scenario you have just been through.

When you have completed the whole set, we can use your choices to create a weighted combination of the ratings. Remember that there is no correct answer; we are only interested in your opinion. Do you have any questions? If not, you may begin when you are ready.

A.12 SAGAT Questions

This section contains the SA questions which have been created and will be given to the participants during the experiments.

SA Question	**Zone**
Which UAV (number) found the object?	Immediate
Where was the person last seen? (Point on map)	Intermediate
How many UAVs are searching for the missing person?	Intermediate
How much time is remaining of completing the zone search	Intermediate
Where is the boat? (Point on map)	Intermediate
What is the temperature of the water?	Long-range
Which direction is the water current?	Long-range
Is the rescue boat in the search zone	Long-range

Table A.9: SA Questions

A.13 Demo Tutorial Introduction

Presentation of the slides in Appendix A.14 to give context to the participant. Then the script below is read to introduce the participant to the SAR interface. When it is expected that the user interacts with the interface, the text will be bold.

1. **System Introduction***You're presented with a multi UAV interface for an operator during a search and rescue operation in the water, with the purpose of searching an area and finding the missing person as fast as possible. The interface can be categorized into three parts that will now be described.*

2. **The Sidebar** The sidebar on your right provides a list of all of the UAVs in the system. Each UAV is tagged with a number in the yellow circle. The circle will change colors between yellow for charging, green for operating and red for a critical error. Further right you will see the current status for the UAV as charging, but it can change between states such as searching and following. Next to that you will be able to see the last update you have received from the UAV, and then next to that you are able to see the current battery percentage. **By clicking** on the UAV card on the right you are able to expand it to get additional information, such as the altitude and location. Last if you **click on the center map**, the map centers around that specific UAV.

3. **The Header** The header in the top provides information about the current mission. By looking at the first card, you will see the current temperature in that area, the wind direction and water currents in the water. This information is generalized for that specific area you are looking at in the map. The second card provides information from the phone call that you have received during the rescue mission. Here you will see when you received the call, and approximately how long the person has been in the water. You will also be provided an estimated search time for the UAVs that are covering the search zone, and a probability of success of finding the person in the water from the various factors. In the right corner you will find two buttons. The first button is for when you receive a mission call that instantiates a search and rescue mission. **By clicking** on the button you will be able to fill out the information that you are given, in this case you may fill in 5 minutes for how long the person has been in the water and a target description of the red jacket. **By saving**, you will see this information in the second card. Lastly, the second button will launch the UAVs and begin the search.

4. **The Map** The map is an overview of all the location specific information. All the numbered circles are a specific UAV, which can be linked to the specific numbered circle in the sidebar. The markers will be updated live during a mission, so you can follow the search. You

can also follow the rescue boat during the operation. Furthermore, you can see a white transparent circle around the rescue boat, this circle is the FOV where it is expected that the rescue boat should be able to see the missing person in. In addition to being able to see the UAVs and the boat, you can see the last known location of the missing person when you received the call. In the interface there exists two kinds of predictive elements of the missing person. Currently in the water you can see a red area, and this is the probability density map of where it is expected that the person may be. However now you see a predictive marker of where it is expected that the missing person has been moving. Both are continuously changing as time goes on.

By now selecting the draw zone tool in the top left corner, you will be able to draw an area that the UAVs will search. When drawing a zone you should try to take into account that the predictive element on the map is a prediction, but it is not exactly where the person may be. When the zone is drawn you have to confirm and save the zone. **You're now able to launch the search** by clicking the Launch UAV's button in the header. In this demo scenario you can see how UAV 7 changes yellow to green and from charging to launching. The moving UAV also has a FOV. It will now start searching for the missing person, and it is expected that the UAV's are not following exactly the search zone that you have drawn, as the algorithm for the paths have to take more into consideration during the search. Now while the boat and UAV is moving, you can follow the operation... What you see now is that the UAV is expecting an action from you as the operator, and this is due to the UAV having detected something in the water. By clicking yes the UAV will stay and follow the object, whereas if you click no the UAV will continue its search. In this case it is not a person, and you should click no to continue the search. **(PAUSE DEMO, SAGOT)**. **(Continue and wait for action)** In this action, the UAV has found the person, and when you select YES the rescue boat will start moving towards the target and your job is complete. It is now expected that you have an understanding of the interface and your instructions as the Operations Manager. If you have any additional questions feel free to ask them now.

A.14 Demo Presentation

Slide 1 - Introduction

Thank you for participating in our user evaluation. This will be a short introduction to familiarize you with your tasks and the process before starting.

User Evaluation

Introduction

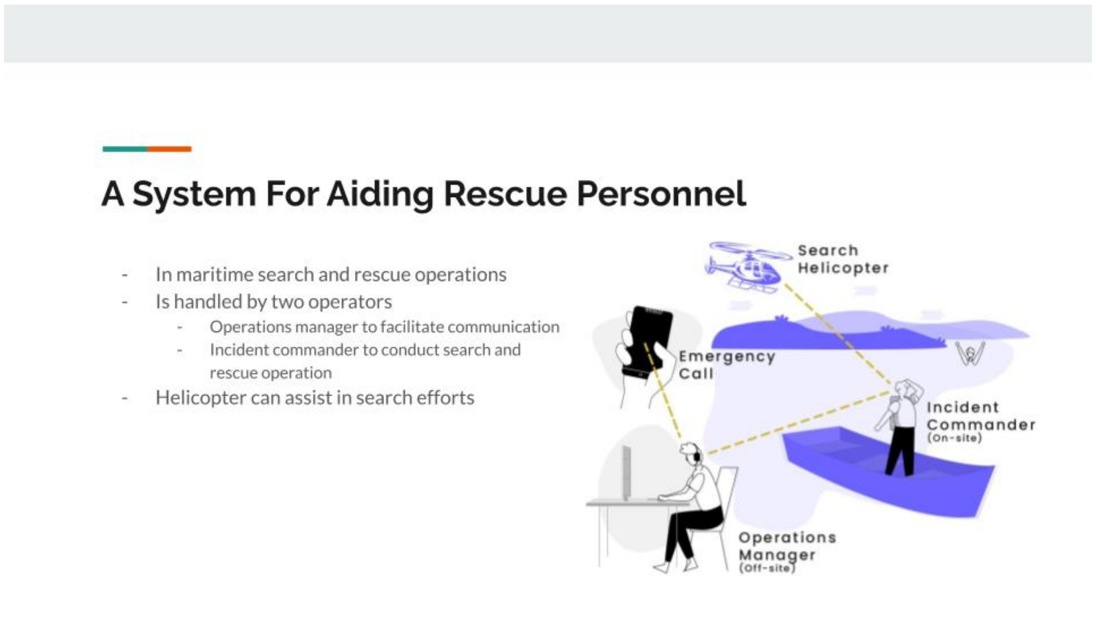
Figure A.18: Presentation slide 1 - Introduction

Slide 2 - Current Rescue Process

You will be evaluating a system which will aid rescue personnel in maritime search and rescue operations.

The figure on the right shows how this is currently done. There are two key personnel during these operations.

- The Operations Manager is there to facilitate communication and gather information. The OM is the one who will receive the initial emergency call.
- The Incident commander is the one conducting the search and rescue efforts on-site, and is in charge of the rescue team. The incident commander and their rescue team will work from the rescue boat during these search efforts.
- In some cases, a search helicopter can requested. This helicopter will mainly be communicating directly with the Incident commander



A System For Aiding Rescue Personnel

- In maritime search and rescue operations
- Is handled by two operators
    - Operations manager to facilitate communication
    - Incident commander to conduct search and rescue operation
- Helicopter can assist in search efforts

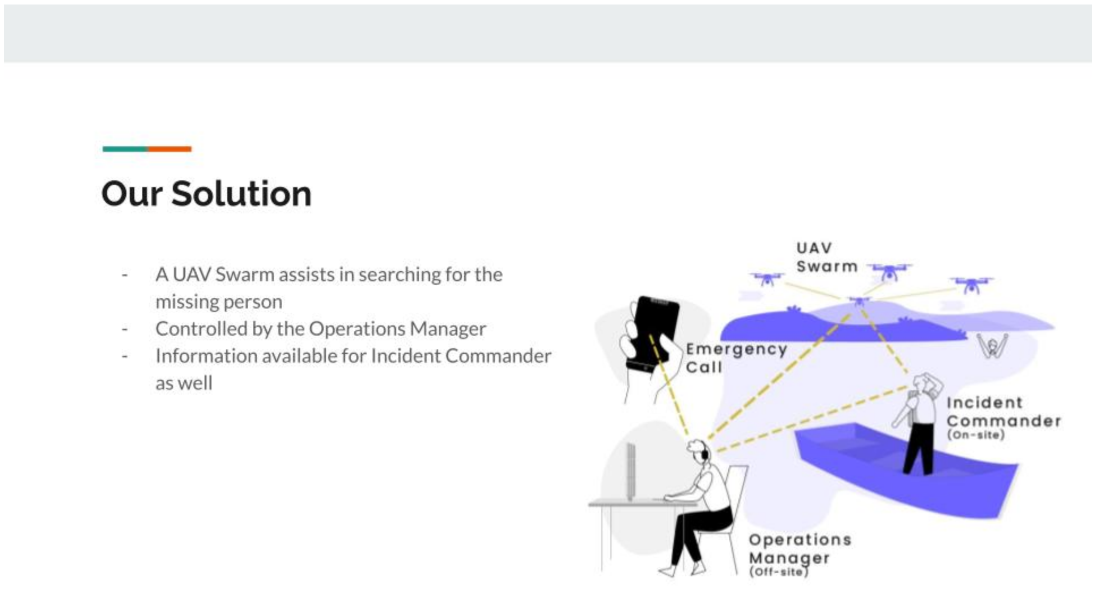
Search Helicopter
Emergency Call
Incident Commander (On-site)
Operations Manager (Off-site)

Figure A.19: Presentation slide 2 - Current Rescue Process

Slide 3 - Our Solution

This is our solution to the rescue process, and is what you will be evaluating.

The search helicopter has been replaced by a UAV swarm, which is controlled and monitored by the Operations Manager. The swarm is mostly autonomous, but some key decisions have to be made. The information from the UAVs will also be available to the Incident commander, but they will not control the system in any way, as they are preoccupied with the search efforts



Our Solution

- A UAV Swarm assists in searching for the missing person
- Controlled by the Operations Manager
- Information available for Incident Commander as well

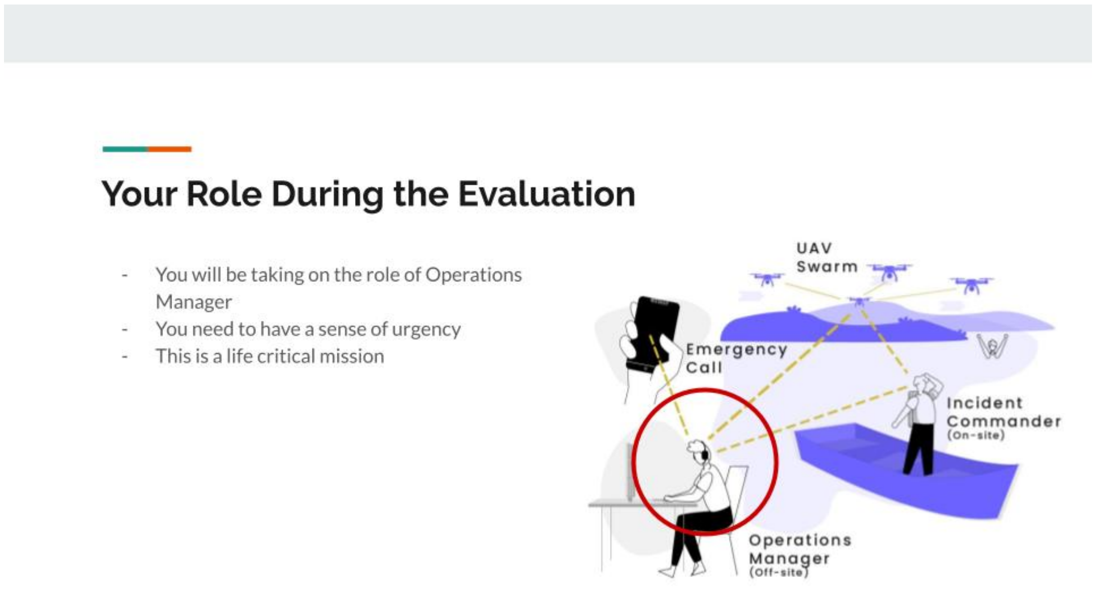
UAV Swarm
Emergency Call
Incident Commander (On-site)
Operations Manager (Off-site)

Figure A.20: Presentation slide 3 - Our Solution

Slide 4 - Your Role

Your role during this evaluation will be that of the Operations Manager. You will control and monitor the system, during a number of simulated rescue operations.

It is important that you have a sense of urgency during these operations, as these are life critical missions.


Your Role During the Evaluation

- You will be taking on the role of Operations Manager
- You need to have a sense of urgency
- This is a life critical mission

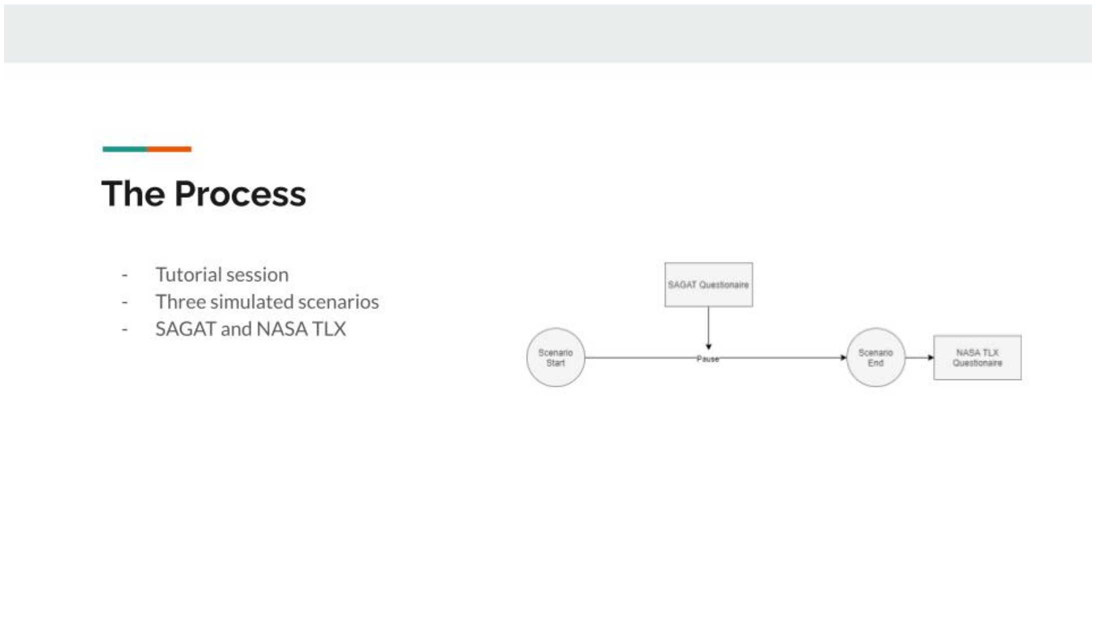
UAV Swarm
Emergency Call
Incident Commander (On-site)
Operations Manager (Off-site)

Figure A.21: Presentation slide 4 - Your Role

Slide 5 - The Process

You will be presented with a tutorial, before the evaluation begins. After this tutorial, we would like you to complete three different simulated scenarios.

During each scenario (and during the tutorial), the simulation will be paused, and you will be given a SAGAT questionnaire, to examine how much you know of the system at the given time. At the end of each scenario, you will be asked to fill out a NASA TLX rating sheet, as well as a NASA TLX weighing sheet. Both of these will be explained after the tutorial has been conducted.



The Process

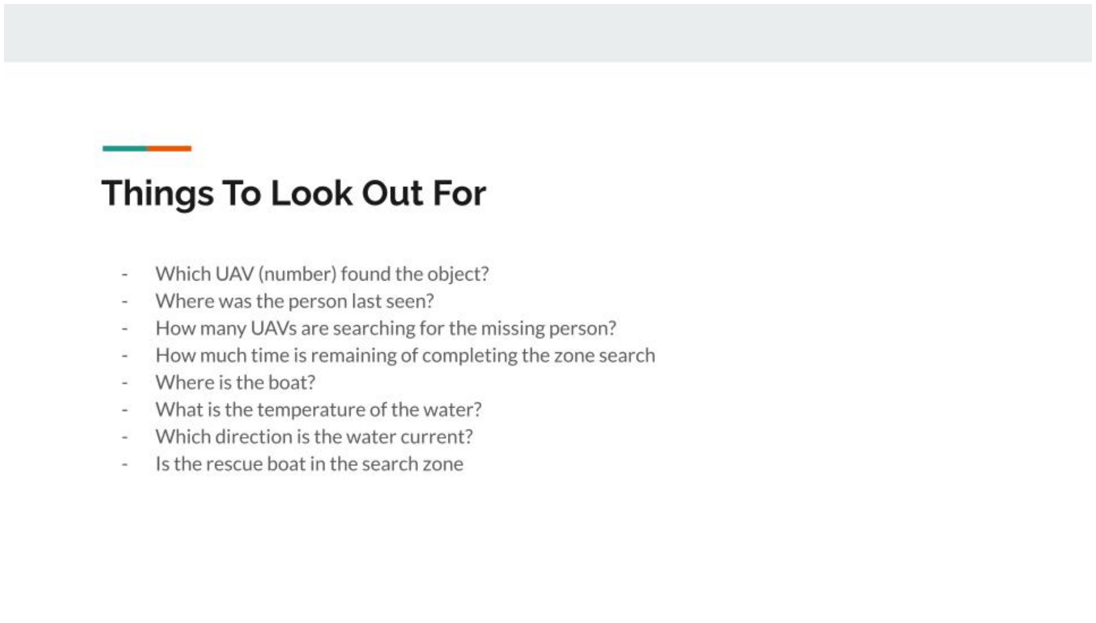
- Tutorial session
- Three simulated scenarios
- SAGAT and NASA TLX

SAGAT Questionaire
Scenario Start
Pause
Scenario End
NASA TLX Questionaire

Figure A.22: Presentation slide 5 - The Process

Slide 6 - Things To Look Out For

Since you are not a expert user of this system, or have any hands-on experience in search and rescue operations, we would like you to be mindful of the following questions, as these might be asked during the SAGAT questionnaire.



Things To Look Out For

- Which UAV (number) found the object?
- Where was the person last seen?
- How many UAVs are searching for the missing person?
- How much time is remaining of completing the zone search
- Where is the boat?
- What is the temperature of the water?
- Which direction is the water current?
- Is the rescue boat in the search zone

Figure A.23: Presentation slide 6 - Things To Look Out For

Slide 7 - Outro

That concludes the introduction. Do you have any final questions before we begin?

Any Questions?

Figure A.24: Presentation slide 7 - Outro

A.15 Consent Form

The purpose of this study is to explore how predictive elements can have an impact on the human factors within human swarm interfaces. The participant will be evaluated on how they perform different tasks.

I understand that I can freely ask questions during the whole study and I understand that I contact the interviewee at any point if any thought occur after the study.

All the data collected will only be used for this study and will not be showing any names or other identifiable characteristics. The data given will be protected and confidential.

Your participation in this study is voluntary and you have the right to withdraw at any time given. There will be no consequences for unanswered questions and the evaluation can be stopped at any point.

I hereby give explicit permission for my demographic information to be used in the project, i.e. age, gender and that selected screenshots or images from the test may be used in the project. Your screen will also be recorded, and used to extract data from the study (only of the interface, not your camera).

☐ Yes ☐ No

I give my consent to participate in the study.

____

Signature of Participant

____

Signature of Interviewee

Email: hci102f21@cs.aau.dk

A.15.1 Interface Changes

This section describes solutions to the interface shortcomings which were found during the pilot study.

Drawing Zone Modal

From the pilot study with our supervisor it was obvious that when the participant had drawn the zone they did not save it. To prevent this it was implemented so that the participant were unable to press the launch search button. Secondly, to also request the user to actually save the zone after it was drawn, a modal were implemented that popup and requests the user to either save or clear the zone drawn.

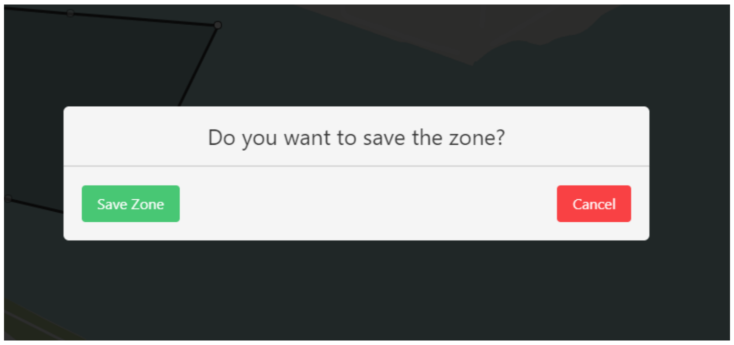
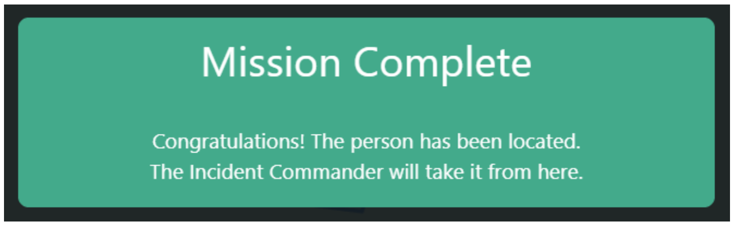

Do you want to save the zone?

Figure A.25: Zone Drawn Modal

As seen in Figure A.25 the participant is requested to take action before continuing, and thus preventing this issue.

Successfully Completion

Another comment received from the pilot study was that the user need confirmation that the scenario was complete. As when the scenario was complete it would just change onto the next scenario, and this could result in the participant lacking the sense of completion. To prevent this a modal were implemented that popup when the boat have successfully found the person. This required a few changes to backend, as the know when the scenario have been complete. The frontend then continuous check if the mission have been complete, and when the mission is complete the modal seen on Figure A.26 is shown.


Mission Complete

Congratulations! The person has been located.
The Incident Commander will take it from here.

Figure A.26: Mission Complete Modal

Full control

One of the biggest issue that was found during the pilot study was that the participant should not be controlling which scenario is selected and what kind of prediction that is shown. This is taking away the focus from the participant, and due to the user study should be conducted virtually the examiner should have full control. This implementation required some additions to the backend, and consisted of two new endpoints for the full control.

The first endpoint is a GET request for checking the current selected parameters, such as the selected scenario and the selected prediction. The second endpoint is a POST request, and is used to select the parameters, such as the selected scenario and the selected prediction. This implementation in the backend meant that in the frontend in the admin panel, it was possible to add a force update button. This force update button would then out from the selected parameter in the admin panel send the POST request to update the selected parameters. The frontend for the participant would then receive this update and would see the changes on his screen. During this the examiner can pause the demo, so that the participant is not seeing any changes.

Small Changes

Further, it was found that the button to named *Launch Search* should be renamed to *Launch UAVs* to make it clear that it is the UAVs that is being launched. The log button from the UAV cards were also removed due to it having no functionality and causing confusion to the participant.

A.15.2 Introduction Changes

This section describes the shortcomings to the introduction which were found during the pilot study. They can be seen below:

- **Set the stage for the participant** It was found that we needed to set the stage for the role of the participant. It was suggested that the current rescue process was explained to the participants, and then present them with our solution afterwards. It was also suggested that we make it very clear which role the participants will have in the evaluation, namely that of Operations Manager. To help set the stage, a set of introduction slides have been created. These can be seen in Appendix A.14.

- **Explain Possibility of Success** The participants are presented with a variable called *Possibility of Success* during the evaluation. This variable is the systems prediction of how likely the mission is to succeed. It was suggested that this variable is explained more clearly during the tutorial introduction. This has been added to the tutorial introduction in Appendix A.13.

- **Generalized Wind and Water Currents** During the pilot study, the participant was unclear whether the wind and water current was for a very specific point or if it was generalized in the entire search sector. This should be made more clear during the introduction. This has been added to the tutorial introduction in Appendix A.13.

- **Guide the Participants in Zone Drawing** Since the goal of the tutorial is to make our participants into capable operation managers, we should make sure to guide them through the process of drawing a good search area, as this is not something they will be experienced with.

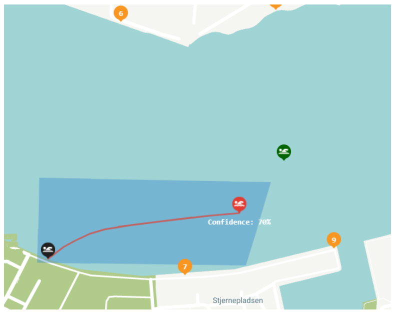
- **Explain How The UAVs work** It should be made clear during the tutorial, how the UAVs function, and that they automatically will cover the search area the participants have drawn.

- **Ask For Questions** Before the evaluation starts, the participants should be asked if they have any questions related to the evaluation. This has been added in the introduction presentation in Appendix A.14.

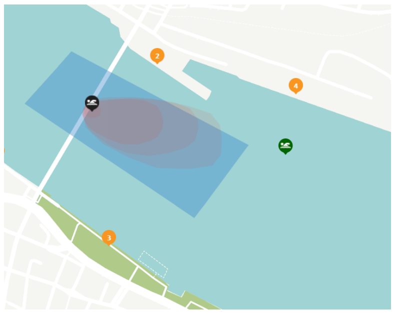
- **Present SAGAT Questions** Since the participants are not search and rescue experts, it should be made clear to them, what they need to be aware of during the evaluation. This can be done by showing the participants the set of questions they will be asked during each scenario. This has been added in the introduction presentation in Appendix A.14.

A.16 Participant Zones

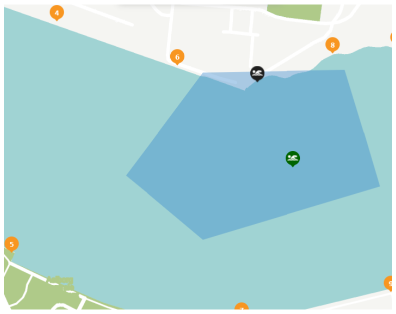
Participants' search zones with the current state of the prediction during the save.



(a) Order 1 - Scenario 1 - Marker

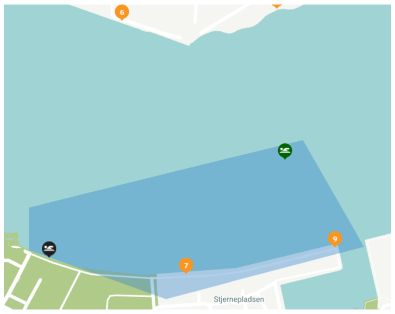


(b) Order 2 - Scenario 2 - Heatmap

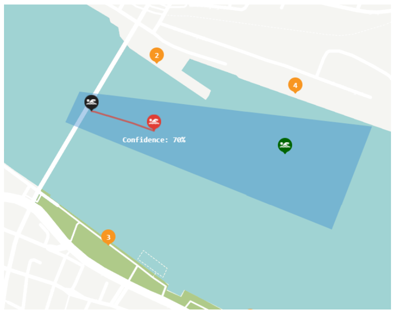


(c) Order 3 - Scenario 3 - Nothing

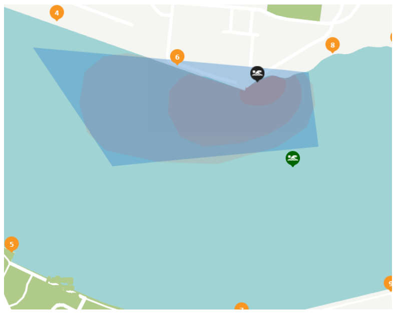
Figure A.27: Participant 1 - scenarios and prediction



(a) Order 2 - Scenario 1 - Nothing

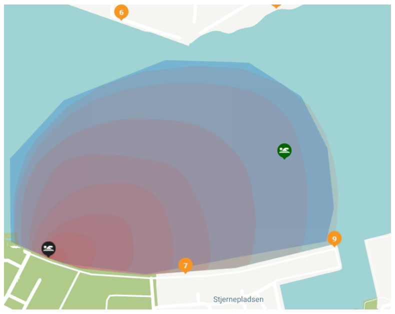


(b) Order 3 - Scenario 2 - Marker

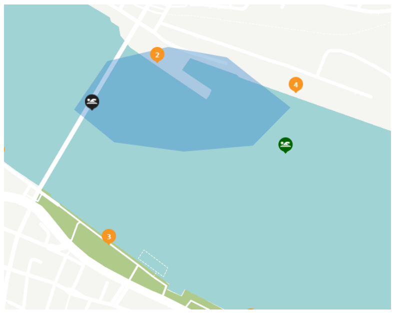


(c) Order 1 - Scenario 3 - Heatmap

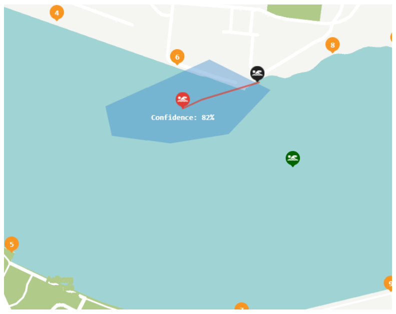
Figure A.28: Participant 2 - scenarios and prediction



(a) Order 3 - Scenario 1 - Heatmap

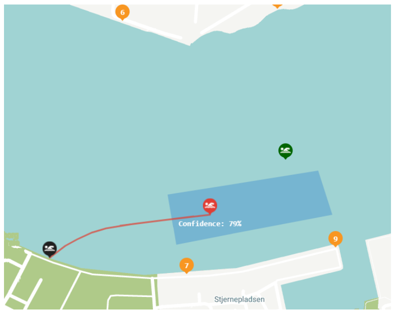


(b) Order 1 - Scenario 2 - Nothing

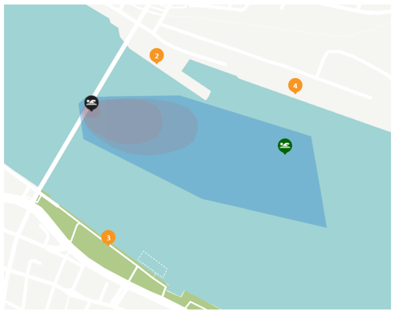


(c) Order 2 - Scenario 3 - Marker

Figure A.29: Participant 3 - scenarios and prediction



(a) Order 1 - Scenario 1 - Marker


(b) Order 2 - Scenario 2 - Heatmap

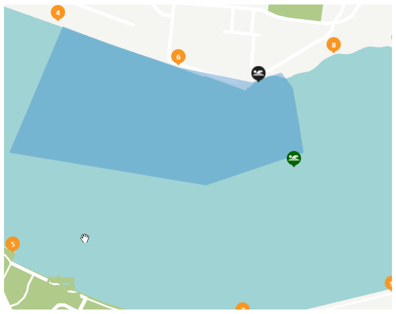
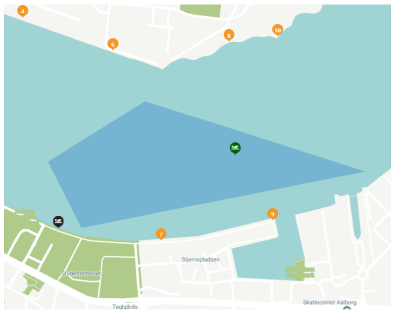
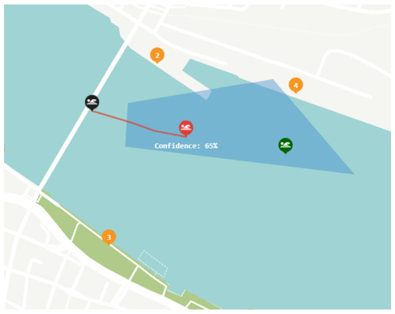

(c) Order 3 - Scenario 3 - Nothing

Figure A.30: Participant 4 - scenarios and prediction



(a) Order 2 - Scenario 1 - Nothing



(b) Order 3 - Scenario 2 - Marker

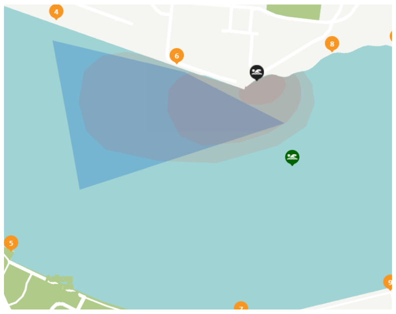
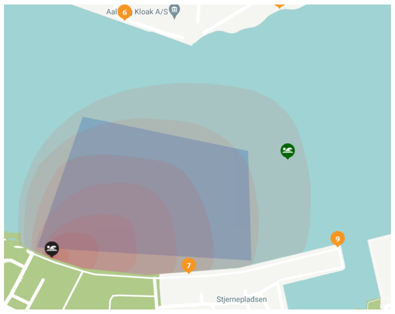
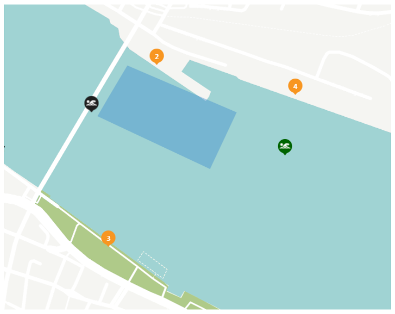
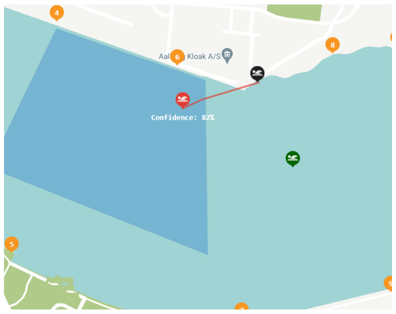

(c) Order 1 - Scenario 3 - Heatmap

Figure A.31: Participant 5 - scenarios and prediction



(a) Order 3 - Scenario 1 - Heatmap


(b) Order 1 - Scenario 2 - Nothing



(c) Order 2 - Scenario 3 - Marker

Figure A.32: Participant 6 - scenarios and prediction

Order	Scenario	Prediction Type	Drawing time	Distance	Zone size
1	1	Marker	22.15s	60m	0.19km²
2	2	Heatmap	14.84s	91m	0.17km²
3	3	Nothing	14.69s	0m	0.30km²

Table A.10: Participant 1

Order	Scenario	Prediction Type	Drawing time	Distance	Zone size
2	1	Nothing	17.69s	0m	0.33km²
3	2	Marker	19.07	0m	0.21km²
1	3	Heatmap	20.32s	45m	0.24km²

Table A.11: Participant 2

Order	Scenario	Prediction Type	Drawing time	Distance	Zone size
3	1	Heatmap	30.00s	0m	0.57km²
1	2	Nothing	22.07s	69m	0.17km²
2	3	Marker	21.6s	173m	0.09km²

Table A.12: Participant 3

Order	Scenario	Prediction Type	Drawing time	Distance	Zone size
1	1	Marker	20.74s	38m	0.08km²
2	2	Heatmap	20.00s	0m	0.20km²
3	3	Nothing	28.13s	26m	0.30km²

Table A.13: Participant 4

Order	Scenario	Prediction Type	Drawing time	Distance	Zone size
2	1	Nothing	20.10s	0m	0.37km²
3	2	Marker	23.91s	0m	0.13km²
1	3	Heatmap	12.90s	109m	0.19km²

Table A.14: Participant 5

Order	Scenario	Prediction Type	Drawing time	Distance	Zone size
3	1	Heatmap	17.19s	96m	0.24km^2
1	2	Nothing	22.15s	150m	0.32km^2
2	3	Marker	19.65s	205m	0.35km^2

Table A.15: Participant 6

Bibliography

A. Agrawal, S. J. Abraham, B. Burger, C. Christine, L. Fraser, J. M. Hoeksema, S. Hwang, E. Travnik, S. Kumar, W. Scheirer, J. Cleland-Huang, M. Vierhauser, R. Bauer, and S. Cox. The next generation of human-drone partnerships: Co-designing an emergency response system. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080.

BMT. Bmt - saris, 1996. URL https://www.bmt.org/bmt-saris/.

D. G. Endsley, Mica R; Jones. *Designing for Situation Awareness An Approach to User-Centered Design, Second Edition*. CRC Press, 2004. ISBN 9781420063554.

M. Endsley. Situation awareness global assessment technique (sagat). In *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*, pages 789–795 vol.3, 1988. doi: 10.1109/NAECON.1988.195097.

Forsvarsudvalget. The annual report of the shipping and aviation rescue council to the sea and airline rescue service in denmark 2019, 2019. URL https://www.ft.dk/samling/20191/almdel/FOU/bilag/112/2234765/index.htm.

B. Gore. Nasa tlx: Task load index, 2020. URL https://humansystems.arc.nasa.gov/groups/tlx/.

hci901e20, J. G. Alsen, R. E. Nielsen, A. H. Jensen, L. Lundbo, L. Rønsholt, and M. T. Boisen. Using UAVs to Aid Decision-Making and Enhance Situational Awareness in Maritime SAR Operations, 2021.

M. Hou, H. Zhu, M. Zhou, and G. Arrabito. Optimizing operator agent interaction in intelligent adaptive interface design: A conceptual framework. 2011. doi: 10.1109/TSMCC.2010.2052041. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-79951955009&doi=10.1109%2fTSMCC.2010.2052041&partnerID=40&md5=ef063493e72c19f968c6a3cc7f473abd.

M. Ilbeygi and M. Kangavari. Comprehensive architecture for intelligent adaptive interface in the field of single-human multiple-robot interaction. *ETRI Volume 40, Issue 4*, 40(4):483–498, 2018.

P. S. Jayam Patel and C. Pinciroli. On multi-human multi-robot remote interaction: A study of transparency, inter-human communication, and information loss in remote interaction. page 44, 2021.

NASA. *NASA TASK LOAD INDEX (TLX)*. Human Performance Research Group, NASA Ames Research Center, Moffett Field. California.

U. S. D. of Homeland Security. *THE U.S. COAST GUARD ADDENDUM TO THE UNITED STATES NATIONAL SEARCH AND RESCUE SUPPLEMENT (NSS) TO THE INTERNATIONAL AERONAUTICAL AND MARITIME SEARCH AND RESCUE MANUAL (IAMSAR)*, 2013.

Regionshospital Nordjylland. Læge i luften, 2015.

G. V. Research. Commercial drone market size, share & trends analysis report by product (fixed-wing, rotary blade, hybrid), by application, by end-use, by region, and segment forecasts, 2021 - 2028, 2021. URL https://www.grandviewresearch.com/industry-analysis/global-commercial-drones-market.

Robotto. Robotto: Saving lives – saving nature, 2020. URL https://robotto.dk/vision/.

J. Roldán, E. Peña-Tapia, P. Garcia-Aunon, J. Del Cerro, and A. Barrientos. Bringing adaptive and immersive interfaces to real-world multi-robot scenarios: Application to surveillance and intervention in infrastructures. 2019. doi: 10.1109/ACCESS.2019.2924938. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85068848361&doi=10.1109%2fACCESS.2019.2924938&partnerID=40&md5=d491c2f38a934a1cac71ed9b17b9d48e.

J. Roldán Gómez, E. Peña-Tapia, A. Martín-Barrio, M. Olivares-Mendez, J. Cerro, and A. Barrientos. Multi-robot interfaces and operator situational awareness: Study of the impact of immersion and prediction. *Sensors*, 17:1720, 2017. doi: 10.3390/s17081720.

J. J. Roldán Gómez. Adaptive and immersive interfaces to improve situational awareness in multi-robot missions, 2018.

M. Ørsborg Johansen. Privat helikopter-eftersøgning kostede Flyvevåbnet 60.000 kr., 2020.