Horizon 2020

Project acronym:	**LASH FIRE**
Project full title:	**Legislative Assessment for Safety Hazard of Fire and Innovations in Ro-ro ship Environment**
Grant Agreement No:	**814975**
Coordinator:	**RISE Research Institutes of Sweden**

LASH FIRE

Deliverable D06.6
Development of smart alert of nearby first responders

August 2022

Dissemination level: **Public**

Call identifier: H2020-MG-2018-Two-Stages Starting date: 2019-09-01 Duration: 48 months
MG-2.2-2018: Marine Accident Response, Subtopic C

Abstract

The sixth work package in LASH FIRE deals with effective manual operations and the aim of Task T06.10 therein, was the development and demonstration of smart alert system of nearby first responders. The research objective was to develop an innovative geo-positioning technology (i.e., longitude, latitude and deck with room level accuracy) to allow more efficient first response to initial fires on ro-ro vessels. To this end, we developed and demonstrate a ground-breaking localization system that requires "zero" fixed infrastructure. Our solution uses static elements of vessel spaces as reference points that can be recognized by commodity smartphone cameras (e.g., deck patterns, bulkhead patterns, hoses, fixed installations, signs, control buttons). The spatial location of vessel objects is collected as a one-off installation process and can then be utilized for technology-driven localization and first response in the early stages of a fire. Besides the core geo-positioning technology, the aim was also to provide the building blocks of a novel vessel indoor information system that will provide fire intelligence during patrol operations. For the above purpose, we developed a fully functional vessel communication software system, coined **Smart Alert System (SMAS),** which integrates besides our localization technology also subsystems allowing first responders to interact over an ordinary network communication channel to exchange messages and data (e.g., heat scans, images, etc.). This provides high levels of situational awareness to cope with information communication bottlenecks in the early stages of a developing fire.

In the context of this deliverable, we provide a complete description of the SMAS software system by building upon the background and related work of D06.4 - Background and testing of smart alert system of nearby responders. Particularly, we start our description out by providing an insight about the layered architecture, where we cover the different software layers that comprise our **SMAS Architecture** at a high level. We also provide more details about the components, languages and technologies used, methodology for unit testing and stress testing. The description provides a clear justification behind the design decision in realizing the SMAS architecture. Subsequently, we provide a structured coverage of the different layers following a top-down approach. We start out from the **Application Layer,** which captures SMAS from the perspective of the user. We particularly cover the following subsystems: *authentication, location, alert, where-am-I, chat and logger*; providing for each subsystem a description, screenshots and the respective application protocol interface (API) endpoints. This explains our software system for the community to capitalize upon our developments but also for future integration tasks. For completeness, we also provide a more detailed coverage of the **Modeling Layer** and **Training Layers,** which are presented in the scope of the overall SMAS software architecture. Finally, we present the **Data Layer** that encapsulates the underlying base tables and views of the SMAS architecture in structured form. We particularly focus on the **Surface algorithm** we designed and implemented in the Structured Query Language to provide infrastructure-less localization using Computer Vision (CV) signals.

To assess the correctness and usefulness of our propositions, we carry out two experimental studies, through collaboration with T05.12 (Stena Flavia ro-pax), namely: (i) a **remote study**, where we collect CV logs for the vessel based on video footage that the operator provided us. We use these logs to carry out a variety of tests and experiments in the laboratory showing a localization accuracy of 80%; (ii) an **on-board study,** where we repeat certain evaluations scenarios on a real drill and compare the quality of results. Our study shows that SMAS is an extremely promising technology that promotes location awareness beyond the current state. Our solutions are developed with open and free technology having no barriers-of-entry and a low cost of operation and maintenance. Additionally, our system is open-source and can both be used for preliminary assessing the benefits of our SMAS system but also for extending the D06.6 outcomes it into a real production system in the future. In conclusion, we believe that our "zero" infrastructure localization technology can play an important role on ro-ro vessels in the complete identification-tracking-positioning spectrum, namely live fire detection and localization, live monitoring, and tactical support, monitoring of cargo on a map, quality control and optimization of cargo load and distribution on ro-ro vessels.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 814975

Document data

Document Title:	D06.6 – Development of smart alert of nearby first responders		
Work Package:	WP06 – Effective Manual Operations		
Related Task(s):	T06.10		
Dissemination level:	Public		
Deliverable type:	R, Report		
Lead beneficiary:	23 – UCY		
Responsible author:	Demetris Zeinalipour		
Co-authors:	Paschalis Mpeis		
Date of delivery:	2022-08-29		
Approved by	Jaime Bleye on 2022-08-29	Jonatan Gehandler on 2022-08-29	Maria Hjohlman on 2022-08-29

Involved partners

No.	Short name	Full name of Partner	Name and contact info of persons involved
23	UCY	University of Cyprus	Demetris Zeinalipour, dzeina@cs.ucy.ac.cy
15	SAS	Sociedad de Salvamento y Seguridad Marítima	Jaime Bleye, jaimebv@centrojovellanos.es
10	STL	Stena Rederi AB	Martin Carlsson, Martin.Carlsson@stena.com

Document history

Version	Date	Prepared by	Description
01	2021-11-25	Demetris Zeinalipour, UCY	Draft Structure
02	2022-07-15	Demetris Zeinalipour, UCY	Draft of final report, circulated to reviewers
03	2022-08-25	Demetris Zeinalipour, UCY	Revision of final report, circulated to reviewers
04	2022-08-29	Demetris Zeinalipour, UCY	Final report

Content

1 Executive summary 6

1.1 Problem definition 6

1.2 Method 6

1.3 Results and achievements 7

1.4 Contribution to LASH FIRE objectives 7

1.5 Exploitation 7

2 List of symbols and abbreviations 8

3 The Smart Alert System (SMAS) Architecture 9

3.1 Introduction 9

3.2 Architecture 9

3.3 Components 10

3.4 Languages and Technologies 11

3.5 Unit Testing and Stress Testing 12

4 SMAS Application Layer 14

4.1 Introduction 14

4.2 User Authentication Subsystem 15
4.2.1 Description 15
4.2.2 Authentication Endpoints 17

4.3 Main Subsystem (Crew Location and Alert) 17
4.3.1 Description 17
4.3.2 Piggybacking Alerts & Message Update Timestamps 19
4.3.3 Report/Get User Location Endpoints 20

4.4 Where-am-I Subsystem 21
4.4.1 Description 21
4.4.2 Localization Endpoints 24

4.5 Geo-Location Chat Subsystem 24
4.5.1 Description 24
4.5.2 Message Send Endpoint 28
4.5.3 Message Get Endpoint 29

4.6 Logger Subsystem 31
4.6.1 Background on Fingerprints 31
4.6.2 Logging UI/UX 32
4.6.3 Logging Focus & Frequency 34
4.6.4 Optical Character Recognition Extension during Logging 34
4.6.5 Logging Upload Endpoint 35
4.6.6 Fingerprint Database Download Endpoint 36

5 SMAS Indoor Modeling Layer and SMAS Training Layer 38

5.1 Indoor Modeling Layer 38
5.1.1 Background on the Anyplace Architect 38
5.1.2 Modeling Vessels in Anyplace 39

5.2 Training Layer 42
5.2.1 Machine Learning Training 42
5.2.2 Training Stena Flavia Classes 42
5.2.3 Managing and Optimizing Models 46

6 The SMAS Data Layer 48

6.1 Database 48
6.1.1 Introduction 48
6.1.2 Database Design 48
6.1.3 User 49
6.1.4 Location 49
6.1.5 Fingerprint (Database) 50
6.1.6 Fingerprint (Query) 52
6.1.7 Object & Object Frequencies Tables 52

6.2 Background on Structured Querying 52

6.3 The Surface CV Localization Algorithm 53
6.3.1 Overview 53
6.3.2 Multiset Subtraction 54
6.3.3 Global Partitioned Frequency Counting 55
6.3.4 Spatial Partitioning of Fingerprints 57
6.3.5 Bounding Rectangle Filtering of Fingerprints 58
6.3.6 The Surface Global (SG) Algorithms 59
6.3.7 The Surface Local (SL) Algorithm 60

7 SMAS Remote Evaluation and Field Study 62

7.1 Experimental Methodology 62

7.2 Remote Study on the Stena Flavia 62
7.2.1 Remote Study Methodology 62
7.2.2 Logging Time Evaluation 64
7.2.3 Scene Localization Accuracy Evaluation: The SG Algorithm 70
7.2.4 Trajectory Localization Accuracy Evaluation: The SL Algorithm 73
7.2.5 Scalability Evaluation 76
7.2.6 Chat Nearest Neighbor Algorithms Evaluation 77

7.3 Field Study on the Stena Vessel 79
7.3.1 Logging Time Evaluation 79
7.3.2 Localization Accuracy Evaluation 80
7.3.3 Battery and Low-Light Evaluation 82
7.3.4 Localization Drill 84
7.3.5 Critical analysis, discussion, and Future ideas 86

7.4 Impact Assessment and Cost Assessment 88
7.4.1 Impact Assessment 88
7.4.2 Cost Assessment 88

8 Conclusion 89

9 References 90

10 Indexes 93

10.1 Index of tables 93

10.2 Index of figures 93

1 Executive summary

Main author of the chapter: Demetris Zeinalipour, UCY

1.1 Problem definition

The LASH FIRE project aims to strengthen the independent fire protection of ro-ro ships by developing and validating effective operative and design solutions addressing current and future challenges in all stages of a fire. In this context, its 6th work package deals with effective fire managing operations and particularly also smart technical solutions for crew geo-positioning to provision efficient first response and effective fighting of fires in their initial stage. The aim of Task T06.10 therein, was the development and demonstration of smart alert of nearby first responders. Particularly, the research objective was to develop an innovative geo-positioning technology to allow more efficient first response to initial fires on ro-ro vessels. Besides the core geo-positioning technology, the aim was also to develop a ship indoor information system and an application-based platform of a ro-ro indoor navigation and indoor fire intelligence system. We develop and evaluate on a ro-ro vessel a technology to localize with room-level accuracy (1-10 meters) with "zero" infrastructure using computer vision localization from deep learning models.

1.2 Method

In D06.6 we present the design and utility of a *Smart Alert System (SMAS)* for quick first response and effective fighting of fires on roll-on / roll-off vessels in their initial stages. Even though in the incident of a fire all crew members may act as first responders, there are some of them (e.g., fire patrol members, able seamen, personnel form the engine control room) that are more likely to act as first responders due to their normal access to restricted cargo decks spaces. Fire patrol members are designated first responders and are the target personnel for the technology we develop in the scope of SMAS.

Equipping first responders with powerful mobile computing devices will allow them to *increase their cyber-physical senses* (i.e., multiple sensing devices like heat scanner in a tiny device), *be connected* (with the bridge and other personnel, discarding possibly outdated communication gear), *be informed* (e.g., carrying bulky manuals and maps in digital form), and *location-aware* (i.e., localization, navigation and tracking of mobile and static assets). These are all dimensions that will increase fire safety by the means of state-of-the-art information technology that has proven itself in everyday life scenarios and that is for the same reason also unobtrusive, with a low learning curve, adaptable through software, and economically viable for massive deployment.

The SMAS architecture builds upon many years of experience with Anyplace [6], which is a Wi-Fi localization, navigation, crowd- sourcing and indoor modeling platform developed over the years at the University of Cyprus. Given that Wi-Fi technology is not widely available on ro-ro vessels, we developed a "zero" infrastructure localization system using Computer Vision (CV) localization.

Particularly, the developed localization technology requires no infrastructure whatsoever (e.g., Wi-Fi, BLE, UWB, RFID, LED) [7,8,9] and it can be deployed at scale with minimal cost on newer or older vessels. Providing a network channel, through sparse Wi-Fi or Mobile 4G/5G antennas can provide the full potential of the developed alerting application and provide a communication channel between first responders. This, however, is only complimentary to the fingerprinting localization technology we develop in this work.

Our proposed method relies on three stages: (i) Training: vessel owners supply to the software team video recordings of the vessel's interior spaces with a particular camera lens focus on static objects. The video recordings are analyzed on a deep learning data center to produce a YOLO (You only look once) state-of-the-art, real-time object detection system Machine Learning model [12] then transformed to a Tensorflow Lite runtime that runs efficiently on a smartphone. This process is carried out once per vessel type (or vessel family - in case multiple vessels have similar internal objects); (ii) Logging: Subsequently, the model is loaded to a smartphone app provided to the vessel owners, which are asked to associate surrounding objects with their location by clicking on a map, yielding a Fingerprint database. Logging is carried out once per unique vessel; and (iii) Localization: The first responders utilize a smartphone application using the fingerprint database that shows both to them and any nearby user (who is connected to a telecommunication network) the location of the responders using the app (e.g., also on the bridge).

1.3 Results and achievements

The aspects are validated through extensive laboratory testing with video traces from Stena Flavia and the laboratory traces. This allowed us to stitch together the technical components that comprise the system architecture of our state-of-the-art "zero" infrastructure localization architecture. Through D06.6 we obtained a complete understanding of the feasibility an infrastructure-free localization method running on commodity smartphone devices by nearby responders.

1.4 Contribution to LASH FIRE objectives

This deliverable contributes to the fulfillment of the LASH FIRE project objective 1 (i.e., validating effective operative solutions) and Objective 2 (i.e., evaluate and demonstrate ship integration feasibility and costs). The technical contributions of this work are summarized as follows:

- We present the developments of a ground-breaking localization technology using computer vision.
- We present the developments of a location-oriented smart alert system relying on computer vision localization, providing quick solutions to a variety of spatial operators.
- We experimentally validate the proposition both through a remote study and an on-site study in the form of a drill.

1.5 Exploitation

The project results can be used by the LASH FIRE consortium and by the shipping community outside of the project in the following manner:

A) **Mobile App:** Usage of the mobile SMAS app developed allowing accurate localization on a vessel with zero infrastructure, showing the vessel plots through a designated Indoor Information System with a tile service (i.e., vessel GIS). This will allow vessel architects and operators to assess the technology in the early stage and get ready for future adoption.

B) **Application Protocol Interface:** Anyplace features a JSON API that allows any technology vendor on the vessel to query and exploit location data from this indoor location service making the system suitable for integration into existing software ecosystems (e.g., FRMC)

C) **Open-Source Software:** All software developments of the Anyplace project are publicly available through Github, particularly https://github.com/dmsl/anyplace. This allows the global community and the LASH FIRE members to assess and benefit of new scientific findings promoting openness.

D) **Publications and Technical Findings:** We published the following introductory [5, 4, 1] and technical papers [13, 14, 15]. We also engaged in COVID-19 actions relevant to navigation [3, 2].

2 List of symbols and abbreviations

FRMC	Firefighting Resource Management Center
CV	Computer Vision
SMAS	Smart Alert System
ML	Machine Learning
APP	Mobile Software Application
GIS	Geographic Information System
IoT	Internet of Things
Ro-ro	Roll-on/roll-off
PAX	Passenger
JSON	JavaScript Object Notation – open data interchange format
UCY	University of Cyprus
IMU	Inertial Measurement Unit
PDR	Pedestrian Dead Reckoning
YOLO	You only look once - state-of-the-art, real-time object detection system
SG	Surface Global Algorithm
SL	Surface Local Algorithm
UI	User Interface
UX	User Experience
POI	Points-of-Interest
CVAT	Computer Vision Annotation Tool by Intel
UCYCO	University of CYprus Objects in Context
LASHCO	LASH FIRE Objects in Context
COCO	Common Objects in Context
NN	Neural Networks
FPN	Floating Point Number
SQL	Structured Query Language
SQL-DML	SQL Data Manipulation Language
SQL-DDL	SQL Data Definition Language
LDAP/AP	Lightweight Directory Access Protocol (Unix) / Active Directory (Microsoft)
PHP	popular general-purpose scripting language suited to web development
KNN	K Nearest Neighbor

3 The Smart Alert System (SMAS) Architecture

Main author of the chapter: Demetris Zeinalipour, UCY

3.1 Introduction

The SMAS (Smart Alert System) is a complete open-source software stack that has been developed in the scope of D06.6 to realize T06.10. SMAS enables first responders to coordinate during patrolling and the first stages of a fire in roro space using only widely available commodity smartphones. SMAS assumes no infrastructure for localization, as it deploys ground-breaking computer vision localization concepts we develop in this work, and only tentatively requires a data communication network necessary for the exchange of digital messages in the scope of its chat.

Even without a communication network, it can still be used by a first responder for localization (i.e., having a powerful digital map and relevant information, expanding the situational awareness of the crew member). The communication network will sooner or later come on a vessel in the form of a few antennas (e.g., 5G or Wi-Fi). What will be missing in this case still, is a vessel referencing system for accurate deck level localization. Current technologies cannot provide room level localization methods and especially with a few antennas have localization accuracy cannot be useful on a vessel. We on the other hand, develop this solution that requires zero infrastructure. In this section we provide additional details about the software architecture of SMAS.

3.2 Architecture

Figure 1 shows the codebases that have been used to bring forward SMAS, summarized as follows:

A) **Anyplace Architect & Anyplace Viewer:** Used to model indoor spaces, i.e., indoor deck plans, Point-of-Interest and Connectors as well the web-based exploration tool (i.e., the viewer). This software has been adapted from our prior work to accommodate vessel spaces.

B) **SMAS (Alert, CV Localization, Authentication, Chat, Logger):** This covers the complete spectrum of components developed in the scope of this deliverable, namely, cv logging and localization, chat, authentication, smart alert.

For the remainder of this deliverable, we focus mainly on LASH FIRE SMAS solely as this has been the focus of this deliverable but survey quickly the rest components.

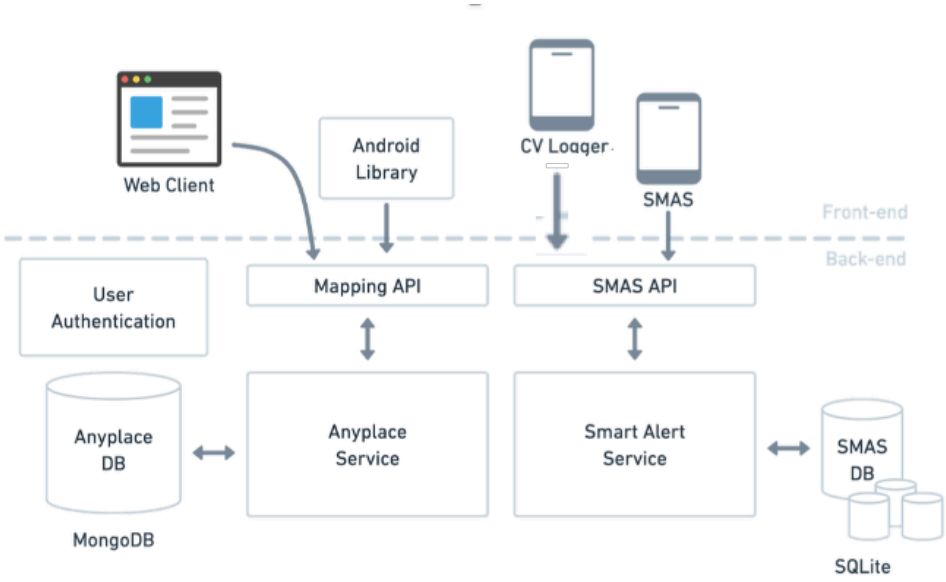


Figure 1: *The SMAS Architecture*

3.3 Components

The SMAS software ecosystem comprises of a variety of components we developed and orchestrated together as presented in ***Figure 2***.

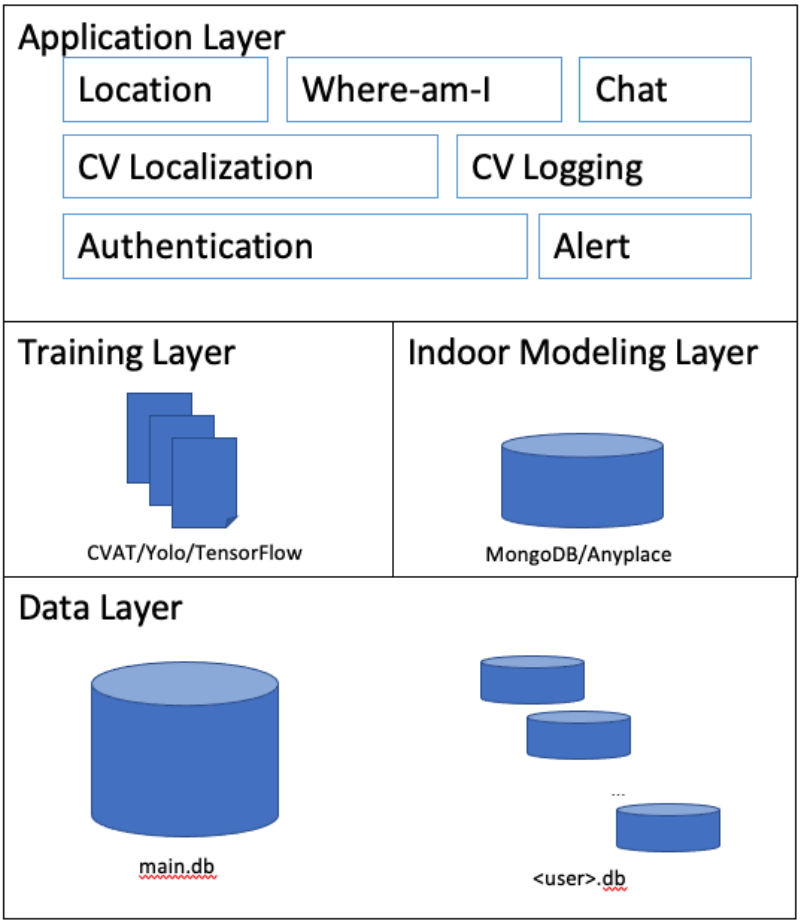


Figure 2: *The SMAS Component Diagram*

We start our presentation out with the Application Layer in Section 4. This layer is the main focus of the D06.6 deliverable as it realizes the solution brought forward from the perspective of the user (i.e., first responder. We provide a very thorough and detailed description so that the system can become useful to other groups and evolve further after the project completion.

For completeness, in Section 5, we augment our description with a summary of the Training and Indoor Modeling layers that have been a primary focus of D06.4 - Background and testing of smart alert system of nearby responders. We continue our description in Section 6 with the Data Layer, which is the place where all relevant data sources of the SMAS architecture are consolidated. The data layer is provisioned to run on the cloud or the edge (e.g., on the vessel control room). We also took relevant actions to allow a partial data layer to execute directly from the handheld of each SMAS user as we aim to offer the maximum usability in the cases of intermittent network connectivity or no network at all.

3.4 Languages and Technologies

We use a wide range of open-source technologies to develop the SMAS architecture as shown in the following list:

Application Layer:

- Client side: Kotlin & Android, Tensorflow Lite
- Server side: PHP and Bash, Linux

Data Layer

- Client side: SQLite
- Server side: SQLite

Training Layer:

- CVAT, YOLO

Indoor Modeling Layer

- Anyplace Architect
- Anyplace Viewer

Our own developments are open to the public to allow maximum exploitation of open new knowledge and societal impact.

3.5 Unit Testing and Stress Testing

In order to allow fine-grain testing of the developed endpoints and system functionality we developed a unit testing framework that allows us to test the correctness of our system across various releases. For this, we developed shell scripts that can simulate connections from the smartphones to our backend service and carry out various testing workloads. We also developed tools to allow simulating workloads of users interacting on our backend architecture to minimize testing time from multiple smartphones.

Using the above logic, we did a thorough stress testing study in Section 7.2.5 that scaled our system to thousands of records and logged cv fingerprints to show that it retains high level of operational readiness

Table 1: SMAS Testing Framework Example used for Unit Testing and Stress Testing

➔ **Testing Login**

```
#!/bin/bash
PAGE_NAME="login.php"

# include common functionality - requires full path for cronjob
source "00-CONFIG.sh"

printf "\n### Login Test: `date`"

MSG_TEST "Incorrect Bearer";
set -x
curl -s -i --compressed -H "Authorization: Bearer aaaaaaaaa"  -H "Content-
Type: application/json" -H "Connection: keep-alive" -H "Accept: */*" -H
"User-Agent: $USER_AGENT" -H "Accept-Language: en-us" -H "Accept-Encoding:
gzip, deflate"  -d '{"uid":"test","password":"1234"}' -X POST $URL
{ set +x; } 2>/dev/null # make set +x not being printed

MSG_TEST "Correct Login";
USER="test"; PASSWORD=`GET_DB_PASSWORD $USER`
RUN_TEST '{"uid":"'$USER'","password":"'$PASSWORD'"}'

MSG_TEST "Incorrect Login";
USER="test"; PASSWORD=`GET_DB_PASSWORD $USER`
RUN_TEST '{"uid":"'$USER'","password":"4321"}'

printf "### DONE\n";

<<COMMENT-OUT
    MSG_TEST "Large Parallel Login Test";
    for i in {0..10000};
    do
        echo $i;
        MSG_TEST "Correct Login";
        USER="test"; PASSWORD=`GET_DB_PASSWORD $USER`
        RUN_TEST '{"uid":"'$USER'","password":"'$PASSWORD'"}' &
```

```
    done
COMMENT-OUT

# print the apache debug log when "-d" is exposed
DEBUG_LOG
# set +xv
```

➔ **Testing Localization**

```
#!/bin/bash
PAGE_NAME="fingerprint-localize.php"

USER="dzeina"; SESSION_KEY=`GET_DB_SESSIONKEY $USER`
MSG_TEST "";

MSG_TEST "LOC(147,178,150):MODELID#3:Algorithm#1";
RUN_TEST '{"uid":"'$USER'","sessionkey":"'$SESSION_KEY'","time":"'`date
+%s`'","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","modelid":3,"algorithm":1,
"cvDetections":[{"oid":147,"height":'$(((RANDOM%999)+1))'.'$(((RANDOM%999)+
1))','width":'$(((RANDOM%999)+1))'.'$(((RANDOM%999)+1))'},{"oid":178,"heigh
t":'$(((RANDOM%999)+1))'.'$(((RANDOM%999)+1))','width":'$(((RANDOM%999)+1))
'.'$(((RANDOM%999)+1))'},{"oid":150,"height":'$(((RANDOM%999)+1))'.'$(((RAN
DOM%999)+1))','width":'$(((RANDOM%999)+1))'.'$(((RANDOM%999)+1))'}]}'

MSG_TEST "LOC(147,178,150):MODELID#3:Algorithm#2";
RUN_TEST '{"uid":"'$USER'","sessionkey":"'$SESSION_KEY'","time":"'`date
+%s`'","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","modelid":3,"algorithm":2,
"cvDetections":[{"oid":147,"height":'$(((RANDOM%999)+1))'.'$(((RANDOM%999)+
1))','width":'$(((RANDOM%999)+1))'.'$(((RANDOM%999)+1))'},{"oid":178,"heigh
t":'$(((RANDOM%999)+1))'.'$(((RANDOM%999)+1))','width":'$(((RANDOM%999)+1))
'.'$(((RANDOM%999)+1))'},{"oid":150,"height":'$(((RANDOM%999)+1))'.'$(((RAN
DOM%999)+1))','width":'$(((RANDOM%999)+1))'.'$(((RANDOM%999)+1))'}]}'
```

4 SMAS Application Layer

Main author of the chapter: Demetris Zeinalipour, UCY

4.1 Introduction

The SMAS application (i.e., SMAS app) is a native AndroidX app and respective backend architecture. The application can execute on any AndroidX capable phone and is written in the Kotlin language. Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets with a massive market penetration and was for this reason chosen for the scope of this project. In theory, the ideas brought forward can be supported on other operating systems as well (e.g., Apple's iOS), even though this will require writing a respective source code in Swift or Objective-C. This is contrary to other type of localization technologies that only operate on older Android devices (e.g., Wi-Fi localization in Android).

For making the SMAS app as responsive as possible and consuming the least number of resources, many functionalities have been implemented in the cloud/edge layer of the SMAS architecture. We do however pay particular attention in retaining the functionality of the primitive underlying operators in the absence of a communication network. Particularly, our app is designed in a way to cope with intermittent and no network connectivity, which means that even when the cloud/edge layer is not accessible due to a network outage or the absence of a network, the zero infrastructure localization functionality will not be lost as we will explain later in this section.

In this section, we provide a detailed outline of the SMAS App UI/UX (User Interface / User Experience). UI refers to the overall visual aspects of a mobile app (e.g., screens, buttons, toggles, icons) while UX refers to the entire interaction you have with the app itself, in terms of internal workflows, interaction design cues and interaction patterns. We designed the Smart App UI/UX by taking into account a requirement analysis with the main stakeholders of the project that insisted o clear and simple design that will allow first responders interact with the system with minimal interaction and input. As such, we took particular attention to optimize input (e.g., voice input when writing messages) or a clear big button for sending out an alert that we will describe in the following sections.

In the subsequent sections, we will analyze the various sections of the SMAS App point-to-point to explain the UI/UX design choices and challenges we were faced with and convey a clear understanding of the product. We also provide all HTTP (Hypertext Transfer Protocol) endpoints in the JSON format.

4.2 User Authentication Subsystem

4.2.1 Description

The first step in using the SMAS app is for a user to authenticate with the SMAS service through the SMAS app. Given that the SMAS service might run on different URLs in the public or private intranet of a prospective deployment, we designed a settings button on the outer layer of the application that allows changing the address of the SMAS service if necessary (the default address is in the cloud on our datacenter).

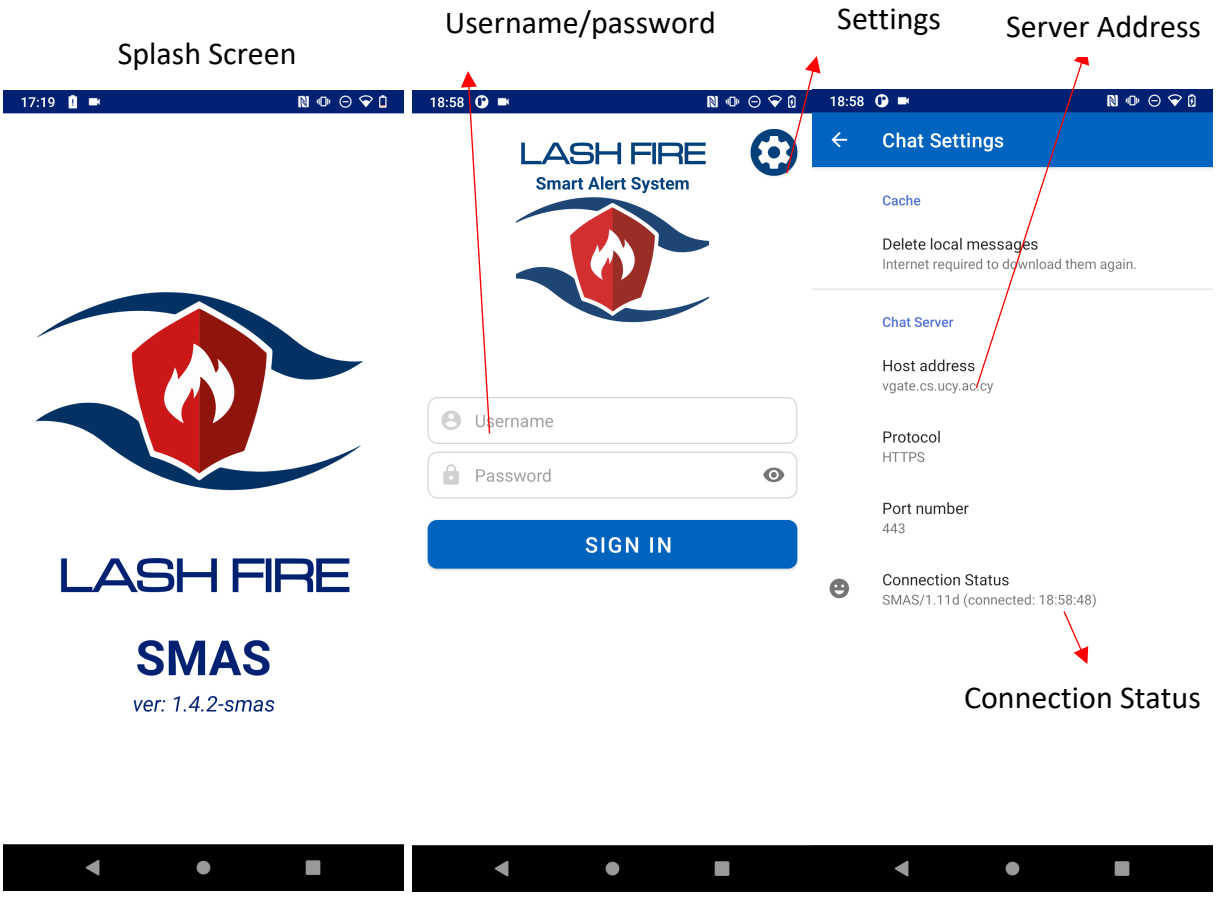


Figure 3: *The SMAS app splash screen (left), user login window (center) and login configuration screen (right) exposing the available functionality to users that are not authenticated yet.*

In respect to security, we utilize different layers to ensure the highest possible security in the given application under the circumstances. Particularly, we provision on the following pillars:

- **HTTPS Communication:** we use Hypertext Transfer Protocol Secure (HTTPS), which is an extension of the Hypertext Transfer Protocol (HTTP) and which is used for secure communication over a computer network. With the https envelope we ensure that hackers on the same network cannot eavesdrop messages between the first responder and the bridge and other first responders. This way, we also shield the system from man-in-the-middle attacks.

- **HTTP Authorization (Bearer Token):** we use this feature to make sure that our underlying communication protocol is only being used by our mobile app and not by bots or hacker on the Internet/Intranet (e.g., carrying out brute-force attacks including denial-of-service attacks over an http-authorization-less protocol). Our http authorization token uses a universally unique identifier that has a length of 128-bit. In case the http-authorization header doesn't match, our protocol will not respond and prematurely preempt the socket communication channel giving hackers less opportunities for interacting with the SMAS service to potentially penetrate it.
- **Session Token & Biometric Authentication:** once a user successfully logs in with the correct username/password we issue a 128-bit session identifier that is active until the next login. This way we minimize the burden of repetitive logins that might be the case on smartphones that have no biometric authorization means (e.g., finger or face recognition disabled). Of course, the latter are automatically granted by the Android OS and are offered over and above the session token mechanism we designed for SMAS users to cover all cases sufficiently. In case a user wants to use the app in offline mode, the user needs to authenticate in an area that has Internet connectivity and then use the application in offline mode (i.e., chat is disabled).
- **Protocol Compression:** we use gzip/deflate sockets to minimize the transfer of data over the communication network. This reduces data transferred over a possibly unreliable and slow communication layer that is usually the case on a vessel.
- **Global Clock:** given that different Android devices might be using their own local time (e.g., different time zones or wrong clock), we offer a global clock synchronization standard as part of every SMAS http response as shown in Table 2.

Table 2: HTTP Response Message from the SMAS Application Protocol Interface.

```
HTTP/1.1 200 OK
Date: Tue, 05 Jul 2022 16:19:34 GMT
Server: Apache/2.4.41 (Ubuntu)
Lash-SMAS-Time: 1657037974
Lash-SMAS-Time-Formated: Jul 05, 2022 19:19:34
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: *
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 96
Keep-Alive: timeout=5
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

4.2.2 Authentication Endpoints

In Table 3 we outline three cases of the SMAS app authentication service that we gracefully handle.

Table 3: SMAS app User Authentication Endpoints

```
### Incorrect Bearer
+ curl -s -i --compressed -H 'Authorization: Bearer aaaaaaaaa' -H 'Content-
Type: application/json' -H 'Connection: keep-alive' -H 'Accept: */*' -H
'User-Agent: SMAS/1.11d' -H 'Accept-Language: en-us' -H 'Accept-Encoding:
gzip, deflate' -d '{"uid":"test","password":"1234"}' -X POST
https://vgate.cs.ucy.ac.cy/smas/login.php

### Correct Login
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"test","password":"1234"}' -X POST
https://vgate.cs.ucy.ac.cy/smas/login.php
{"status":"ok", "uid":"test", "sessionid":"C2B6FCC5-DD8F-4B35-B785-
21A79CABC66B"}

### Incorrect Login
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"test","password":"4321"}' -X POST
https://vgate.cs.ucy.ac.cy/smas/login.php
{"status":"err", "uid":"test", "descr":"wrong uid or password"}
```

4.3 Main Subsystem (Crew Location and Alert)

4.3.1 Description

The first aspect when designing the app was to immediately convey to first responders a clear understanding of the location of other first responders on a given vessel. We assume that all first responders use the SMAS app using the localization technology we describe in the subsequent sections. In ***Figure 4***, we present the main screen of the app. In terms of interaction with other first responders we provide the following functionality that will be detailed more in the next sections:

- **Map interface & Floor Selector:** The basic elements of the design include a map interface of the vessel with a respective deck selector that allows a user to freely move around the vessel maps to search for specific Points-of-Interest. By moving to a specific deck, a first responder can see the location of other members at each deck. The map and the POIs can be registered through the Anyplace Web Interface as explained in D06.4 - Background and testing of smart alert system of nearby responders.
- **Location of Other First Responders:** a first responder can obtain quick situational awareness of the latest reported location of first responders in the system as these appear on the map

interface automatically. By clicking on the interactive yellow dot, one can obtain additional information such as who is the given person and what time the last location was reported. In case the location of a fellow first responder is older than a given timepoint we record a different dot color explained next.

- **Where-am-I:** A user can click the Where-am-I button that allows instant engagement of the computer vision location system and identification of the location on the screen with a blue dot relative to the location of other first responders in yellow.
- **Send Alert:** This button sets the given first responder into "ALERT" mode and will display a respective alert on all other first responders.
- **Chat:** Here the first responder can interact with other first responders with instant messages.
- **Heat Camera:** enables the FLIR camera to obtain heat images that can be shared on the Chat or for the first responder to investigate a given area of concern. To use this functionality, a first responder needs to use a smartphone that has an embedded heat camera (like the Caterpillar S62 and S61 we use in T06.10). In the future, we also aim to investigate the automatic scanning of heat images to provide a notification to the first responder for additional actions.
- **Settings:** These provide numerous settings for tuning the system to variety of user preferences we will cover in the subsequent sections.

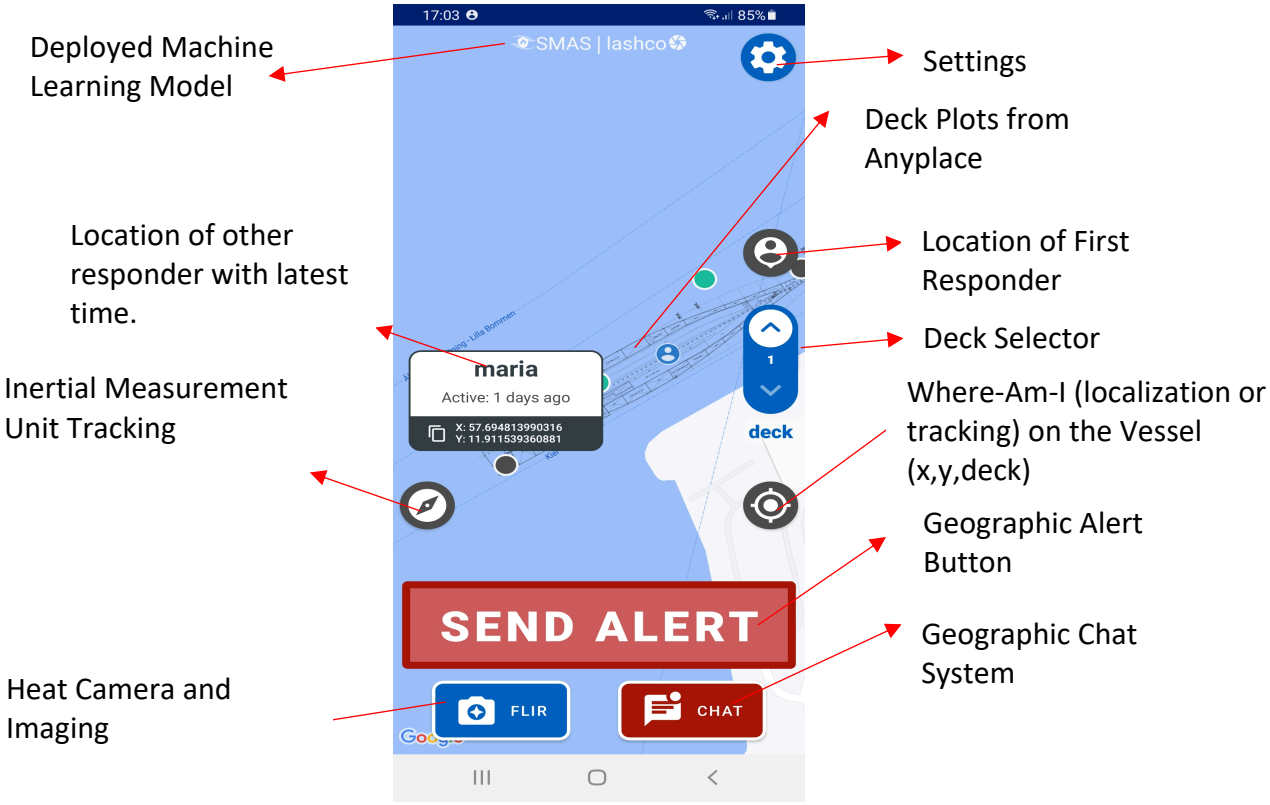


Figure 4: The SMAS app main screen showing the various functionalities available to the first responder.

This first and albeit most important subsystem of SMAS is the first responder location subsystem, which provides the indoor location of a first responder to both the responder him/herself (i.e., for location awareness) but also to fellow first responders and the bridge (the app can run on the bridge using an Android Emulator, like BlueStacks available on Windows, macOS and Linux). This subsystem is responsible to present the last know location of first responders on the map on all devices used by the system. The SMAS indoor location of a first responder (by the means of CV localization longitude, latitude, and deck number), are consolidated on the edge/cloud in a central database and conveyed to participating users based on parameter we provide in the settings of the SMAS app (default: every 2 seconds).

In Section 4.3.3 we present the respective endpoint that shows how the system accepts besides the (x,y,deck) location information also the timestamp, vessel id and whether or not the given user is in alert mode, so in summary (vesselid, timestamp, x, y, z, alert). The given locations are rendered on the mobile device every X milliseconds, a parameter that can be adjusted in the settings we will present next. The results are shown on ***Figure 4*** as yellow dots and ***Figure 6*** as a green dot.

4.3.2 Piggybacking Alerts & Message Update Timestamps

We use piggybacking of the alert mode **(alert)** and message update timestamps **(msgts)**, which is the technique of carrying a message along with the location update, to conserve precious network resources and minimize communication.

- In case a user is in **alert mode,** this information will be sent to the central database and be fetched on the devices of other first responders in an automatic manner as part of this protocol. This will respectively initiate an auditable alert on the device of receiving first responders and provide additional information (such as location of the alert) that can be used by first responders to either engage in clarification over the chat or rush to the location of the first responder in need for additional help.
- In case a user is in **chat mode,** a user's smartphone might or might not pull additional messages from the service to conserve network resources based on a timestamp we return from the server.

4.3.3 Report/Get User Location Endpoints

Figure 5 shows the location of a first responder using the CV localization mode and the manual mode. We also show the location of other responders (green for recent reports and gray for older reports). The above coloring scheme gives a clear location awareness to the first responders team and the bridge in case of an emergency. In ***Figure 5*** we present the various colors used for the location balloons. In Table 4 we outline the two main endpoints of the SMAS User Location Report and Get endpoints.

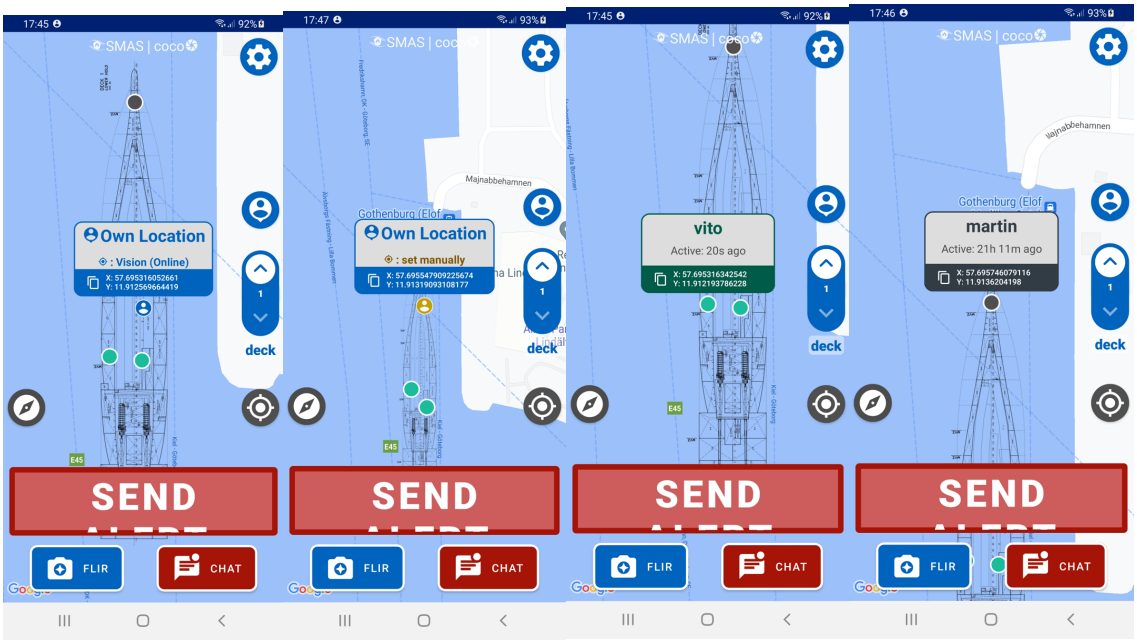


Figure 5: *The SMAS app showing the location of various first responders with different colors (blue: active cv location, orange: manual location, green: active location of other responder, gray: inactive location of other responder.*

Table 4: SMAS User Location Report and Get endpoints

➢ **Report User Location Endpoint**

```
curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection: keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d '{"uid":"vito","sessionkey":"61ACE202-32DF-4E3F-839C-77AE700F8738","time":"1657039055","buid":"vessel_2a2cf77c-91e0-41e2-971b-e80f5570d616_1635154314048","x":57.6952026025419716,"y":11.91225098622780165218,"deck":1,"alert":0}' -X POST https://vgate.cs.ucy.ac.cy/smas/location-send.php
{"status":"ok", "uid":"vito", "rows":1}
```

➢ **Get User Location Endpoint**

```
### Location Get Test: Wed Jul  6 09:19:06 UTC 2022
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection: keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-Language: en-us' -H 'Accept-
```

```
Encoding: gzip, deflate' -d '{"uid":"dzeina","sessionkey":"E167ACD8-DBAE-4F03-9249-
232F5F126AD7"}' -X POST https://vgate.cs.ucy.ac.cy/smas/location-get.php
{"status":"ok", "uid":"dzeina",
"rows":[{"uid":"dzeina","time":1657092379,"timestr":"Jul 06, 2022
10:26:19","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.694847855553,"y":11.911533325911,"deck":3,"alert
":0,"msgts":1657031388,"servertime":"2022-07-06
09:19:05","name":"Demetris","surname":"Zeinalipour"},{"uid":"jaime","time":16570991
43,"timestr":"Jul 06, 2022 12:19:03","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695251122542,"y":11.912184516228,"deck":1,"alert
":0,"msgts":1657031388,"servertime":"2022-07-06
09:19:05","name":"Jaime","surname":"Bleye
Vicario"},{"uid":"maria","time":1657099143,"timestr":"Jul 06, 2022
12:19:03","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695273052542,"y":11.912324036228,"deck":5,"alert
":0,"msgts":1657031388,"servertime":"2022-07-06
09:19:06","name":"Maria","surname":"Hjohlman"},{"uid":"martin","time":1657099141,"t
imestr":"Jul 06, 2022 12:19:01","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695292622542,"y":11.912245436228,"deck":1,"alert
":0,"msgts":1657031388,"servertime":"2022-07-06
09:19:06","name":"Martin","surname":"Carlsson"},{"uid":"paschalis","time":165643062
7,"timestr":"Jun 28, 2022 18:37:07","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695249039745,"y":11.912561282516,"deck":3,"alert
":0,"msgts":1657031388,"servertime":"2022-07-06 09:19:06","name":"Paschalis
","surname":"Mpeis"},{"uid":"vito","time":1657099143,"timestr":"Jul 06, 2022
12:19:03","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695101962542,"y":11.912295816228,"deck":1,"alert
":0,"msgts":1657031388,"servertime":"2022-07-06
09:19:06","name":"Vito","surname":"Radolovi\u0107"}]}### DONE
```


4.4 Where-am-I Subsystem

4.4.1 Description

This subsystem enables the camera system to identify the surrounding objects of a user upon which our novel *Surface* algorithm is executed to find the location on a map (i.e., longitude, latitude, and deck). The operation of the functionality at a high level is presented in ***Figure 6***. The resulting location is presented as a green dot on the map and enables both the first responder to obtain location awareness but also other first responders likewise. The details of the Surface algorithm will be presented in Section 6.3.

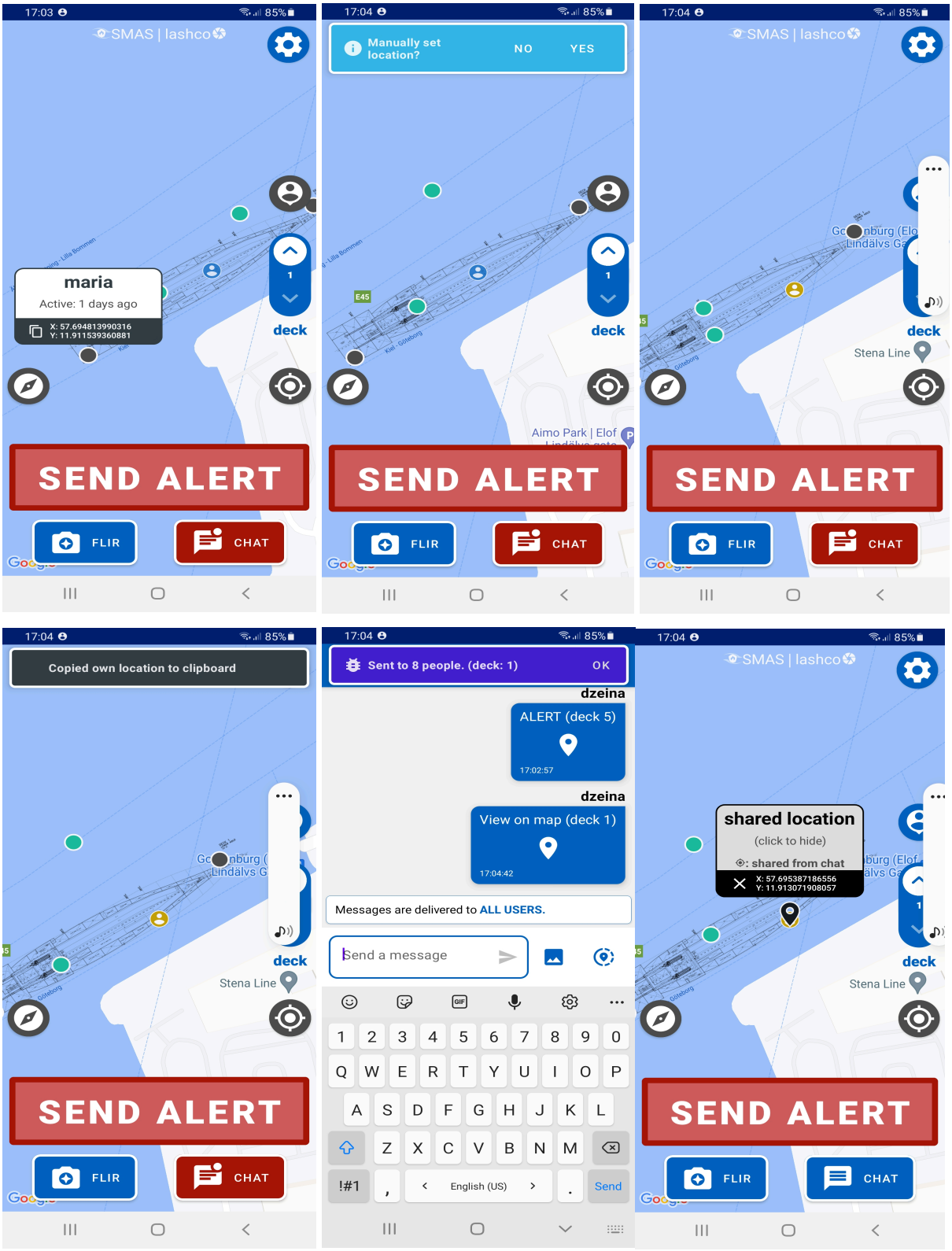


Figure 6: *The SMAS app First Responder Location; requesting the Where-I-Am functionality enables the camera system in the background to obtain the latest user location; which is subsequently presented on the map along with the location of other first responders. Sharing the location of a user becomes seamless as a long-press is all it takes to share any location on the vessel on the chat channel or some other medium (e.g., email).*

In this section we discuss the localization subsystem that will be discussed in more detail when we outline the *Surface* localization algorithm in Section 6.3. At a high level, we can think that different types of objects recognized by the camera-system are organized in a geolocation database we coin the SMAS Fingerprint Database. The objective of the *Surface* algorithm is to identify the row that is most similar to the current object set of a user. In order to illustrate this, consider the example in Table 5.

Table 5: Example Fingerprint Localization Scenario

Query (Current Fingerprint):

{**object-5, object-10, object-99**}

Fingerprint Database:

Row #1: (**x-1, y-1, deck-1,** **object-5, object-10**, object-100)

Row #2: (**x-2, y-2, deck-2,** object-11, **object-5**, object-78)

….

Row #N: (**x-N, y-N, deck-N,** object-15, object-15, **object-99**)

Query Result:

(**x-1, y-1, deck-1**, **object-5, object-10, object-100**)

The objective of the Where-am-I subsystem is to identify the closest match of the query to the Fingerprint Database. The query doesn't clearly contain the location but the fingerprint database has the location (x,y,deck) associated with each set of objects as this has been carried out during the Logging operation we will describe in Section 4.6. On the example, we can clearly see that Row #1 contains the most similarity to the query as both object-5 and object-10 are found (with object-100 not agreeing on object-99). Row #2 and the rest rows don't have any closer match, as such Row #1 is returned as the closest (x,y,deck) location to the user.

Let's now have a look at the implementation of the above basic idea in the SMAS system and look later at the specifics of the localization algorithm. For the example below, we provide, beyond the object identifier also the dimensions of the object that provide an indication on how far/close an object is to the user obtaining the location. Regarding the response from the service (or the local smartphone – in case we carry out offline localization), we also observe that the service returns a dissimilarity score that provides an indication to the app how good the location estimate is and that we will discuss again later.

4.4.2 Localization Endpoints

In Table 6 we outline the localization endpoint with model #2 (i.e., UCYCO). We will explain all options and parameters later in Section 6.3.

Table 6: SMAS Localization Endpoint

```
curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-A57F-
A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection: keep-
alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-Language:
en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"dzeina","sessionkey":"A3CC1D38-041C-4E45-AD1D-
8506E0557C63","time":"1657899708","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","modelid":1,"prevX":57.695298672318,"prevY":11.
912288032472,"prevDeck":4,"cvDetections":[{"oid":3,"height":443.414,"width"
:354.755},{"oid":3,"height":700.558,"width":733.370},{"oid":4,"height":541.
269,"width":923.788},{"oid":18,"height":347.769,"width":692.559},{"oid":24,
"height":912.370,"width":946.457},{"oid":24,"height":706.929,"width":64.48}
,{"oid":24,"height":634.721,"width":739.8},{"oid":69,"height":522.136,"widt
h":114.840},{"oid":69,"height":942.258,"width":293.632}]}' -X POST
https://vgate.cs.ucy.ac.cy/smas/fingerprint-localize.php


{"status":"ok", "uid":"dzeina", "sql_time_msec":26.963949203491,
"rows":[{"flid":55,"x":57.695546475805,"y":11.912978030741,"deck":5,"xDiff"
:0.0002478034869994872,"yDiff":0.0006899982689994033,"deckDiff":1,"dissimil
arity":4,"weight":0.014423076923076924}]}
```

4.5 Geo-Location Chat Subsystem

4.5.1 Description

The SMAS Geo-Location Chat subsystem is an instant text messaging application that has been developed in the scope of this deliverable to perfectly align to the utility philosophy of SMAS as well as the respective UI/UX principles in terms of menus, colors, visual cues, and interaction patterns. In ***Figure 7***, we present the main components of the SMAS Geo-Location Chat, which comprises of the toolbar on the top, a message wall where both text messages, multimedia objects (e.g., images), location pins and alert messages can be circulated by first responders. The given system can be reached from the main screen of the app by clicking the chat button.

The SMAS Geo-Location Chat system provides along with each message the indoor location (i.e., longtitude, latitude and deck floor) making this a one-of-a-kind instant messaging application for vessels and separating it clearly from the wealth of instant-messaging apps that are readily available on the mobile phone markets (e.g., WhatsApp, Facebook Messenger, WeChat, QQ Messenger, Telegram, Viber, Line, and Snapchat). Currently, no other instant messaging app [10,11] can provide

the location on the vessel within a few meters as we do with our innovative computer vision localization system developed in T6.10.

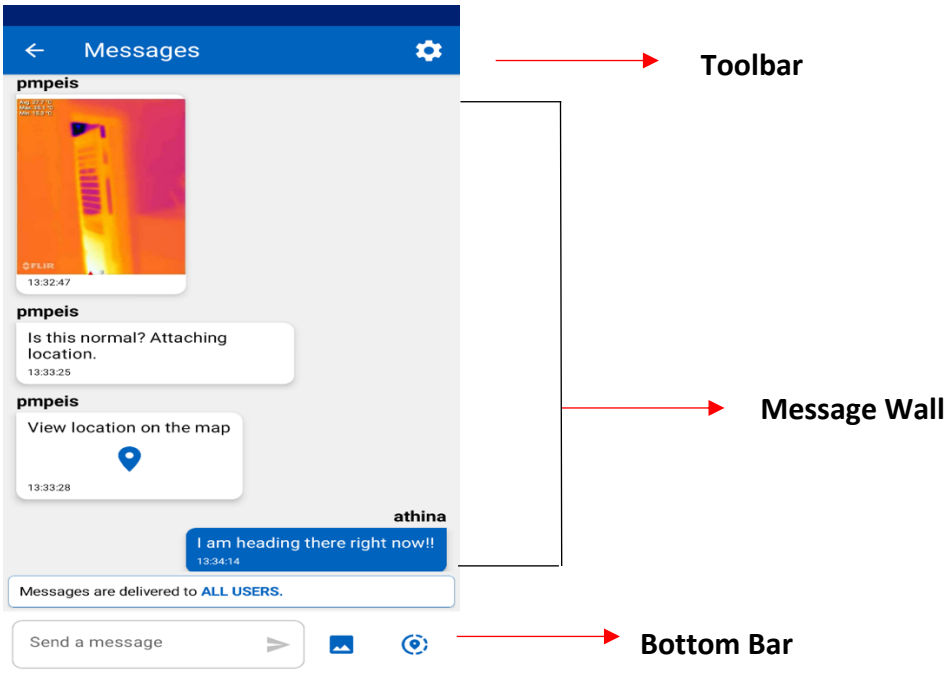


Figure 7: *The SMAS Chat system enables first responders to exchange instant messages in real time to share mission critical information to fellow responders and the bridge on incidents of concern (e.g., fire in the initial stages).*

In the subsequent paragraphs, we explain the underlying functionality and protocols of the chat app. The first operation is to present the messages of other users on the message wall. For this we have developed the following options that are combined on the request to create a variety of functionality as explained in the subsequent paragraphs:

Sender Statuses (mtype):

1. Regular Geo-Location Message
2. Geo-Location Message with Attachment
3. Share-My-Location Geo-Location Message
4. Alert Geo-Location Message

Recipient Statuses (mdelivery):

1. All users
2. Users on the same deck
3. Closest-k users (system parameter $smas_db_messages_knn)
4. Users within a bounding rectangle (system parameter $smas_db_messages_bound_meters)

To demonstrate a message sending operation, let us consider the scenario shown in ***Figure 8***. Here first responder jaime sends out at alert to all users on the chat channel in ***Figure 8***a. The given message is picked up by user dzeina in ***Figure 8***b who responds that he is approaching as he is nearby. When

arriving, dzeina takes a heat scan of the vehicle in question and sends it on the chat channel to the rest first responders. User martin opens the image on his smartphone and carries out a magnification as shown in ***Figure 8***c and ***Figure 8***d. This provides user martin with additional details about the incident before coordinating with the bridge for subsequent actions or before rushing to the incident site.

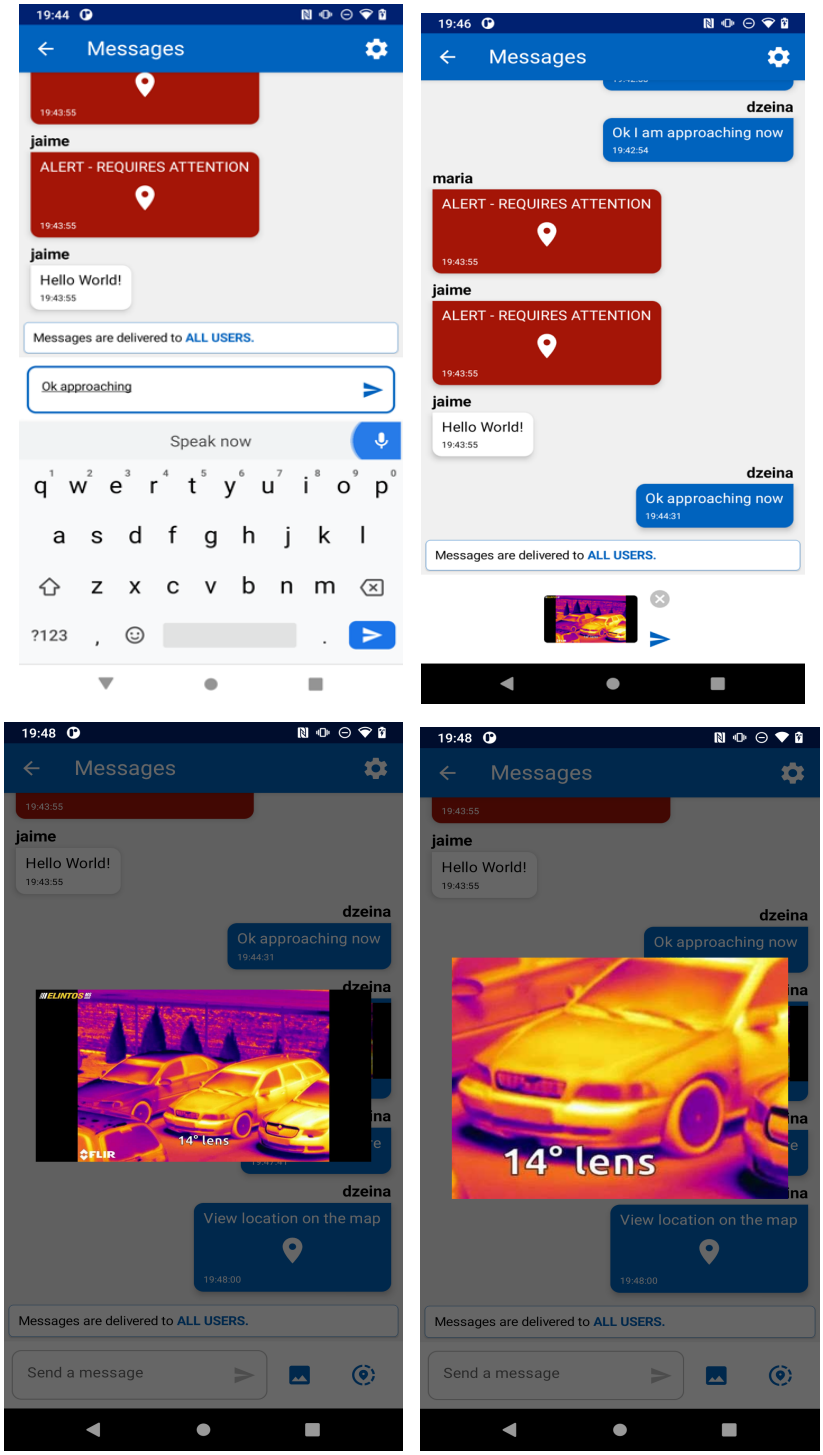


Figure 8: *The SMAS Geo-Location Chat subsystem allows first responders to interchange location-based messages text messages with the assistance of a) voice recognition, b) multimedia attachments, and c-d) attachment zoom and magnification functionality.*

As a next action, let us take a closer look at options supported through the chat system. In ***Figure 9***a, user dzeina shares the location of the incident through the "Share Location" functionality. This is like what a user would obtain on location-based system only now that this provides accurate room-level localization on the vessel space within a few meters. Additionally, this provides the deck level of the incident immediately and without additional hardware, where elevation can only be obtained otherwise on smartphones with the provision of a barometric sensor and then again mapping this to the vessel deck is not straightforward. As such, our solution fills this gap in the localization spectrum.

In ***Figure 9***b, we observe another very useful functionality that has to do with the prospective recipients of a message. By default, all users on a given SMAS server will receive the communication. This might not be practical in scenarios involving several tens of users. As such, we designed a message delivery filtering method (mdelivery), which allows a sender to specify who should receive a given message. For the purpose of distance calculation we use the Euclidean distance and x,y,z, given that z is an integer.

In ***Figure 9***c, we show how the chat button becomes red in cases a message is automatically pulled in the background by the SMAS service as part of the message piggybacking mechanism (msgts) presented in Section 4.3.1. One final remark is that attachments are encoded in base64 encoding to enable maximum compatibility with different byte ordering architectures (big endian vs. little endian). This adds a little bit in size but is counter-acted by the fact that our protocol uses gzip sockets.

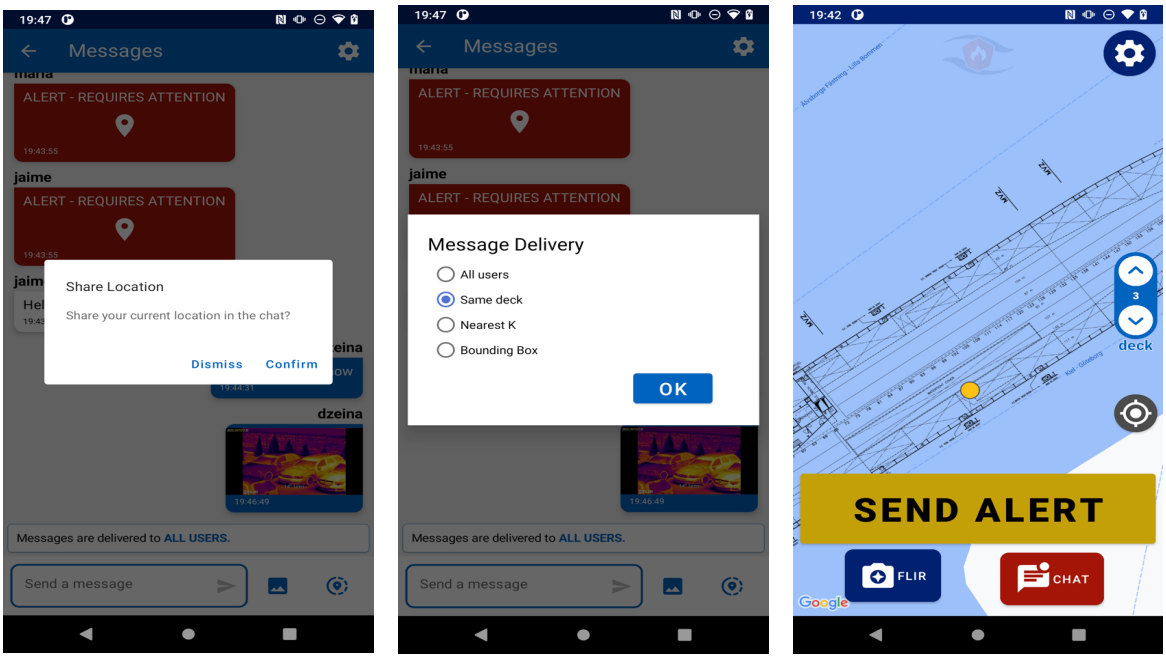


Figure 9: *The SMAS Geo-Location Chat features the following functionality: a) location share; b) preference selectors to define recipients (all users, same deck users, k-nearest and users within a bounding rectangle); and c) visual cues to allow a first responder know that a text message has been received.*

4.5.2 Message Send Endpoint

In Table 7 we outline the chat endpoint with various options.

Table 7: SMAS Geo-Location Chat Send Subsystem API

```
### Send Message (mtype#1) to ALL (mdelivery#1)
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"jaime","sessionkey":"4F7703AB-3A18-4526-9CDB-
93E0C42BD0F7","time":"1657129251","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.69571425419716,"y":11.912741462278016521
8,"deck":1,"mtype":1,"msg":"Hello World!","mdelivery":1}' -X POST
https://vgate.cs.ucy.ac.cy/smas/msg-send.php
{"status":"ok", "uid":"jaime", "x":57.695714254197, "y":11.912741462278,
"deck":1", rows":8}
### Send Message (mtype#1) to DECK (mdelivery#2)
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"martin","sessionkey":"5ABF8523-E8AC-4CFE-A4E7-
A773A09D8F9C","time":"1657129251","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.6953017125419716,"y":11.9122811622780165
218,"deck":1,"mtype":1,"msg":"Hello World!","mdelivery":2}' -X POST
https://vgate.cs.ucy.ac.cy/smas/msg-send.php
{"status":"ok", "uid":"martin", "x":57.695301712542, "y":11.912281162278,
"deck":1", rows":4}
### Send Message (mtype#1) to KNN (mdelivery#3)
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"maria","sessionkey":"FB4E2003-9B9A-4F3A-9D5D-
E4F88B95D7D4","time":"1657129251","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695918625419716,"y":11.912117462278016521
8,"deck":1,"mtype":1,"msg":"Hello World!","mdelivery":3}' -X POST
https://vgate.cs.ucy.ac.cy/smas/msg-send.php
{"status":"ok", "uid":"maria", "x":57.69591862542, "y":11.912117462278,
"deck":1", rows":3}
### Send Message (mtype#1) to BB (mdelivery#4)
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"pmpeis","sessionkey":"3E4E1592-DFE4-48F3-A1EE-
A499192D55AD","time":"1657129251","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695978025419716,"y":11.91221078622780165
218,"deck":1,"mtype":1,"msg":"HelloWorld!","mdelivery":4}' -X POST
https://vgate.cs.ucy.ac.cy/smas/msg-send.php
{"status":"ok", "uid":"pmpeis", "x":57.69597802542, "y":11.912210786228,
"deck":1", rows":0}
### Send Attachment (mtype#2) to KNN (mdelivery#2)
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"martin","sessionkey":"5ABF8523-E8AC-4CFE-A4E7-
A773A09D8F9C","time":"1657129251","buid":"vessel_2a2cf77c-91e0-41e2-971b-
```

```
e80f5570d616_1635154314048","x":57.6951851025419716,"y":11.9123183162278016
5218,"deck":1,"mtype":2,"msg":"PCFET0NUWVBF.....hcHQtc2l6ZS1hZGp1bWw+","me
xten":"jpg","mdelivery":3}' -X POST https://vgate.cs.ucy.ac.cy/smas/msg-
send.php
{"status":"ok", "uid":"martin", "x":57.695185102542, "y":11.912318316228,
"deck":1", rows":3}
```

Besides the Send Message Functionality, our system also supports the Get Message Functionality that comes with the following options.

Get Statuses (mgettype):

0. All messages (this starts on purpose at value 0)
1. Last-N messages
2. Specific Message by ID (mid)
3. Messages since a timestamp (time)
4. Messages between 2 timestamps (time)
5. Messages of a specific user (uid)
6. Messages of a specific message type (mtype)

All messages are limited by a system-wide Input/Output size reduction parameter named $smas_db_message_block to minimize the burden on the wireless communication channel if necessary. The default configuration in the SMAS client is message type #3, based on the piggybacked parameter msgts discussed earlier in Section 4.3.1.

4.5.3 Message Get Endpoint

In Table 8 we present the Geo-Location Chat Get Messages API call exposing how the introduction of the “Messages since a timestamp (time)” option can reduce total communication by over 500x times. Similarly, using the other endpoints can provide application developers quicker and more efficient access to the stored data.

Table 8: SMAS Geo-Location Chat Get Messages API

```
### mgettype=1. Fetching all messages is expensive but sometimes necessary
(e.g., after app reset)
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"martin","sessionkey":"5ABF8523-E8AC-4CFE-A4E7-
A773A09D8F9C","mgettype":1}' -X POST https://vgate.cs.ucy.ac.cy/smas/msg-
get.php

{"status":"ok", "uid":"martin",
"rows":[{"uid":"jaime","time":1657129251,"timestr":"Jul 06, 2022
20:40:51","buid":"vessel_2a2cf77c-91e0-41e2-971b-
```

e80f5570d616_1635154314048","x":57.695714254197,"y":11.912741462278,"deck":1 …

➔ **SIZE: 526787 Bytes**

mgettype=3. Fetching the latest messages is the method we use in the SMAS chat app as it is lightweight.

```
curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection: keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d '{"uid":"martin","sessionkey":"5ABF8523-E8AC-4CFE-A4E7-A773A09D8F9C","mgettype":3,"from":1657129251}' -X POST https://vgate.cs.ucy.ac.cy/smas/msg-get.php
```

{"status":"ok", "uid":"martin",
"rows":[{"uid":"jaime","time":1657129251,"timestr":"Jul 06, 2022 20:40:51","buid":"vessel_2a2cf77c-91e0-41e2-971b-e80f5570d616_1635154314048","x":57.695714254197,"y":11.912741462278,"deck":1,"mid":"236D7721-2BF3-456C-9E01-6E329CFB6DE3","mtype":1,"msg":"Hello World!","mexten":"","mdelivery":1},{"uid":"martin","time":1657129251,"timestr":"Jul 06, 2022 20:40:51","buid":"vessel_2a2cf77c-91e0-41e2-971b-e80f5570d616_1635154314048","x":57.695301712542,"y":11.912281162278,"deck":1,"mid":"B2F977F7-9958-4DDB-8A82-60AFF331ECF7","mtype":1,"msg":"Hello World!","mexten":"","mdelivery":2},{"uid":"jaime","time":1657129251,"timestr":"Jul 06, 2022 20:40:51","buid":"vessel_2a2cf77c-91e0-41e2-971b-e80f5570d616_1635154314048","x":57.69532162542,"y":11.912272226228,"deck":1,"mid":"18CF54EE-595C-4B0E-95A3-4F5FDDB360A6","mtype":4,"msg":"","mexten":"","mdelivery":1}]}

➔ **SIZE: 889 Bytes**

4.6 Logger Subsystem

4.6.1 Background on Fingerprints

The SMAS Logger Subsystem is a mobile application we developed in AndroidX that is responsible for the collection of spatially annotated object-sets necessary for the computer vision localization we developed. Let us consider the following abstract example in Table 9 to explain the utility of a fingerprint and the respective fingerprint database at a high level.

Table 9: Example of the SMAS Computer Vision (CV) Fingerprint

Single Fingerprint:
(**x, y, deck, object-5, object-10**, object-100)

These fingerprints present the underlying data source for the task of localization. We can think about the above record being collected by the means of a crowdsourcing task where the installer of the localization system walks around on the vessel with the SMAS Logger to collect object-sets and associating them with a map (i.e., latitude x, latitude y, deck) by long-pressing on the map interface. These fingerprints are then organized in a central database shown in Table 10, creating the notion of a **Fingerprint Database.** Although we will elaborate more closely on the structure of the Fingerprint Database in Section 6, understanding the high-level structure at this point is important for the interpretation of the user interface.

Table 10: Example of the SMAS CV Fingerprint Database

Fingerprint Database:
Row #1: (**x, y, deck, object-5, object-10**, object-100)
Row #2: (**x, y, deck,** object-11, **object-5**, object-78)
….
Row #N: (**x, y, deck,** object-15, object-15, **object-99**)

The Fingerprint database physically resides on the cloud/edge server and can from there be used for the task of localization as shown in Section 6.3. Of course, the given fingerprint database is also transferred the users of the SMAS apps at prespecified intervals or on-demand to enable users to localize in cases of no network or intermittent network connectivity issues that might be predominant especially in older vessels.

4.6.2 Logging UI/UX

The SMAS Logging subsystem is implemented in AndroidX to perfectly align to the utility philosophy of SMAS as well as the respective UI/UX principles in terms of menus, colors, visual cues, and interaction patterns. In ***Figure 10***, we present the main components of the SMAS Logger, which comprises of the toolbar on the top, a scan button that starts the logging as well as a Where-am-I button used for quick verification of the contribution of a fingerprint to the overall localization accuracy improvement. As we can observe in the sequence of screenshots, a user can enable the logger through the settings menu and then collect fingerprints for specific areas of interests. Collection of signals can either be carried out in front of a computer (which is what we call ***remote logging***) or in the actual environment (which is what we call ***on-board or in-person logging).*** In the scope of D06.6 we will carry out both remote logging and on-board logging to expose the pros and cons of each of these methods and compare the results.

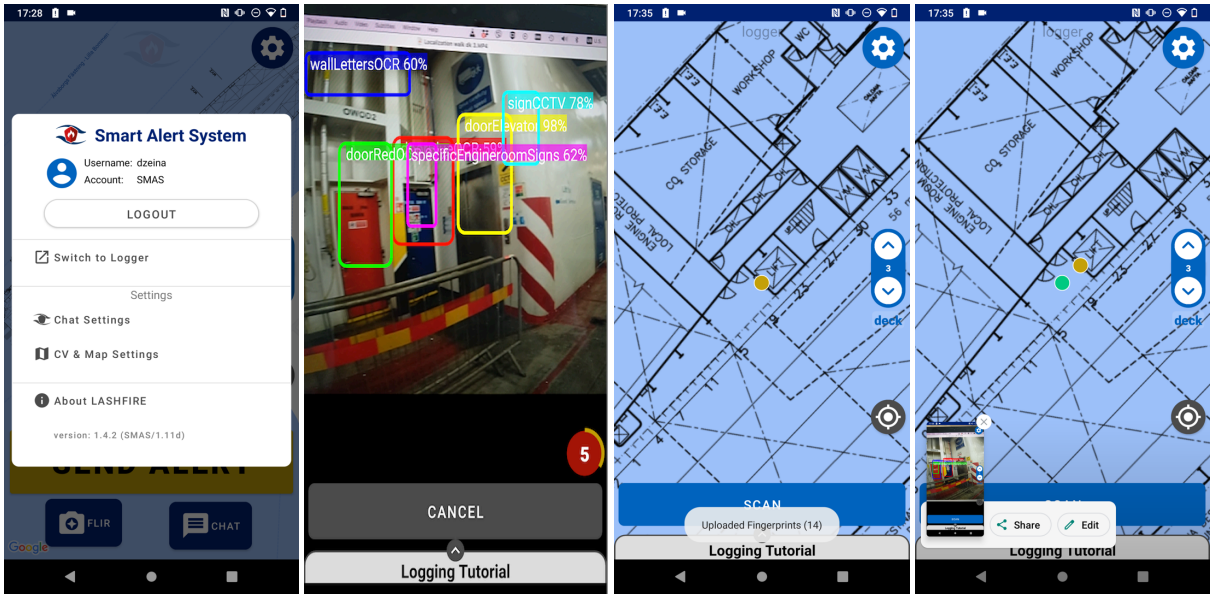


Figure 10: *The SMAS Logger subsystem enables the installers of the localization service to associate locations on a vessel (longitude, latitude and deck) with the surrounding objects. (a) The Logger is enabled through the main settings menu, where a user can switch between the main SMAS app and the SMAS Logger; (b) When clicking the Scan button the SMAS Computer Vision subsystem starts collecting surrounding objects for a user defined interval (a user can cancel or discard a collection if necessary); (c) The fingerprints are uploaded to the fingerprint database (in case of intermittent connectivity stored locally until the cloud layer is accessible); (d) a logger can instantly hit the "Where-am-I" button to see if how the collected signals improve the localization accuracy at the given location or whether the collection of additional scans is necessary.*

The SMAS Logger also allows an installer to quickly go through the logging guidelines and familiarize him/her self with the logging process. In ***Figure 11***a, we show the guidelines slide bar that an installer can slide any time necessary. Even though we named as installers some remote team, the localization setup personnel can be crew members or first respondents that will crowdsource the vessel object

surfaces following these simple guidelines and then benefit from accurate localization. This has a tremendous impact on deployment costs as in theory setup can be carried out by the vessel personnel itself in case a cost reduction is required (i.e., self-deployment).

Finally, the SMAS logger is equipped with a variety of machine learning models. An installer can switch through these models with a simple menu option as shown in ***Figure 11***b. Developing specialized models for new vessels is possible through the workflow described in D06.4 - Background and testing of smart alert system of nearby responders. In the future we aim to look at a complete lifecycle for developing, evolving and maintaining these computer vision models and linking them to the CV localization system we developed.

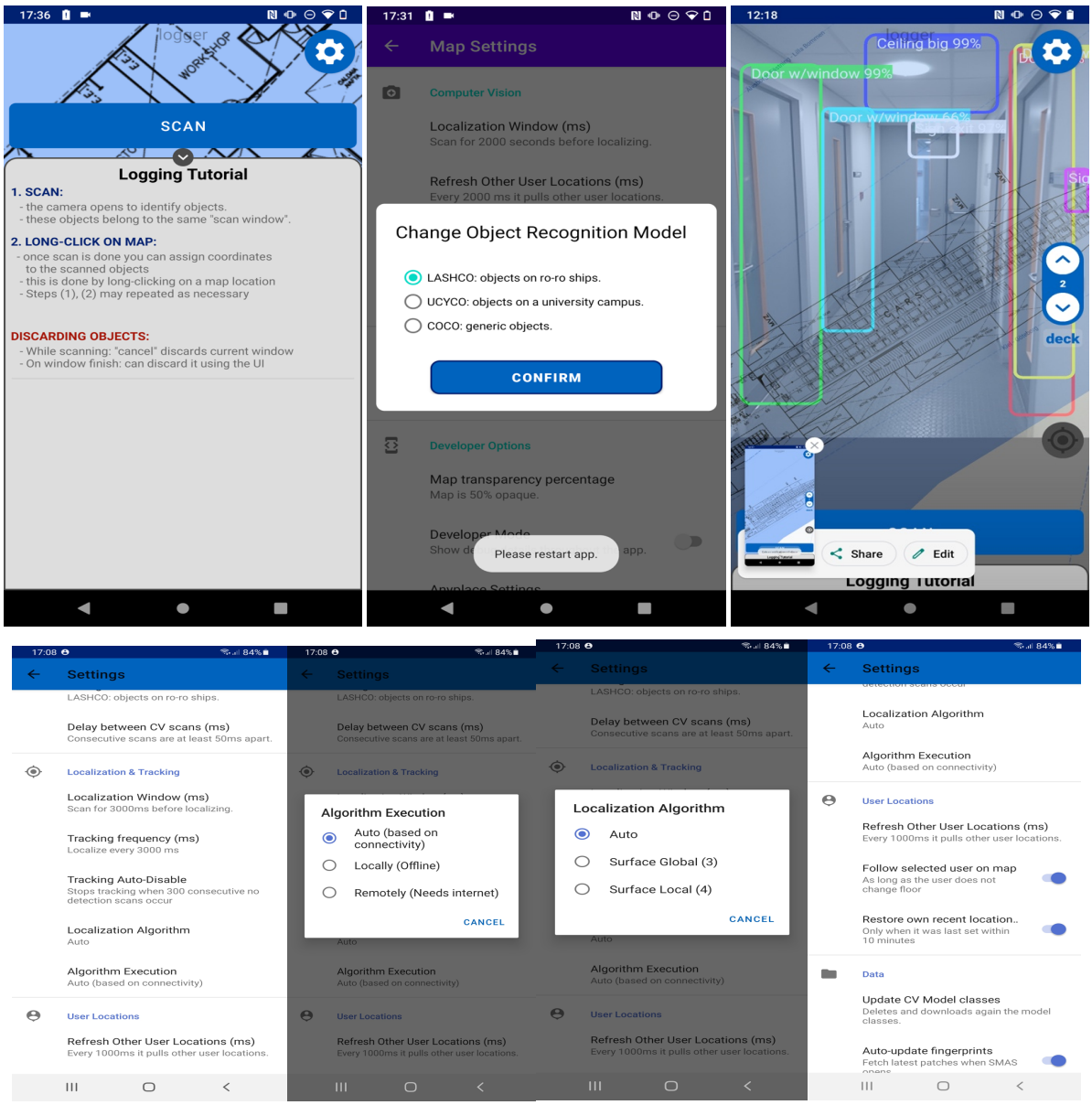


Figure 11: *The SMAS Logger guidelines and settings. A variety of options enable battery optimization, caching and performance. An installer can also enable debug messages and select between online/offline localization and the localization algorithm. These are important for mission critical scenarios and to expose the capabilities of SMAS with Zero infrastructure.*

4.6.3 Logging Focus & Frequency

Logging is carried out once per vessel at installation time but can be repeated over time in case new static objects are added to the environment (e.g., a new EV charger). Logging focuses on static environment assets (doors, stairs, elevators, signs, ev chargers, ceilings, floor patterns, etc.) that have been incorporated in the computer vision machine learning model we constructed during the training phase. The objective is to collect for various (x,y,deck) on the vessel from different angles and locations the surrounding objects such that Surface algorithm we designed and implemented in this project can localize users indoors with high accuracy and zero infrastructure (i.e., without any infrastructure whatsoever, namely Wi-Fi, BLE, UWB, RFID, Sonar, LED).

4.6.4 Optical Character Recognition Extension during Logging

Optical character recognition (OCR) is the electronic or mechanical conversion of images of typed, handwritten, or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo. In the case of SMAS Loggers, we use off-the-shelf OCR libraries to refine the logging process. Particularly, all classes carrying the OCR label, e.g., BlueDoorOCR, are candidates to run through the below additional step. Note that the YOLO inference engine chops up a full image into smaller squares for all these squares it runs inference. In ***Figure 12***b, we show one such a sample that has been produced for LASHCO recognized object BlueDoorOCR in ***Figure 12***a.

Figure 12: *Carrying out OCR on selected surfaces in the SMAS Logger provides greater resolution on the type of object identified (e.g., in the example we know that we have the STB Pilot door bunker and not any door but there is also additional autocorrection potential of the wrong captured words through the AndroidX SpellCheckerService).*

After ***Figure 12***b passes through an OCR library it returns the following results:

- com.google.android.gms.vision result (deprecated):
  STB PILOT DOO STELON
- Google's ML Kit available in numerous languages (https://developers.google.com/ml-kit/guides) result:
  STB PILOT DOOR BUNKER TTION'

The above shows clearly how a simple BlueDoorOCR object can now be differentiated as a specific type of door (an STB PILOT DOOR BUNKER). The complete set of objects passed on to the OCR recognition engine are shown in Table 9. In the scope of the SMAS Logger we use the Google ML-Kit, for text recognition. It is on-device (offline), real-time text recognition, with a small application footprint. It can separate words/elements, lines and even paragraphs (link: https://developers.google.com/ml-kit). Had we also deployed spell-checking services and libraries on Android, again in multiple languages, makes the task of refining the preliminary OCR a promising venue for future refinements.

Table 11: List of Objects extended with OCR technology

wallFireExtinguisherEmbeddedOCR wallLettersOCR wallNumberOCR doorBlueOCR	doorRedOCR signLetterOCR wallSingleDigitOCR wallCapWhiteOCR signNumberOC

4.6.5 Logging Upload Endpoint

In Table 12 we show how different objects are logged using our respective endpoints. In fact, we have implemented a very sophisticated unit testing and stress testing environment that captures application state directly from the SQLite database to make access legitimate given the multiple layers of security we added to the authentication mechanism.

Table 12: Uploading SMAS Logging Data to the SMAS Service

```
### Fingerprint Send: Thu Jul  7 13:27:08 UTC 2022

### Log 3-object fingerprint: 178, 150, 147
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"martin","sessionkey":"5ABF8523-E8AC-4CFE-A4E7-
A773A09D8F9C","time":"1657200428","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.6951943625419716,"y":11.9129088622780165
218,"deck":1,"modelid":3,"cvDetections":[{"oid":178,"height":350.1966,"widt
h":927.8066},{"oid":150,"height":347.0995,"width":447.76758},{"oid":147,"he
ight":186.02942,"width":135.67444}]}' -X POST
https://vgate.cs.ucy.ac.cy/smas/fingerprint-send.php
{"status":"ok", "uid":"martin", "rows":3}

### Log 2 same objects fingerprint: 178, 178
```

```
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"martin","sessionkey":"5ABF8523-E8AC-4CFE-A4E7-
A773A09D8F9C","time":"1657200428","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.6951790625419716,"y":11.9121407762278016
5218,"deck":1,"modelid":3,"cvDetections":[{"oid":178,"height":350.1966,"wid
th":927.8066},{"oid":178,"height":347.0995,"width":447.76758}]}' -X POST
https://vgate.cs.ucy.ac.cy/smas/fingerprint-send.php
{"status":"ok", "uid":"martin", "rows": 2}
```

4.6.6 Fingerprint Database Download Endpoint

In Table 13 we outline the fingerprint database download. We will explain how these fingerprints create the Fingerprint Database in Section 6.1.5.

Table 13: Uploading SMAS Fingerprint Database Download

```
###
### DB Struct Get Test: Thu Jul  7 15:39:39 UTC 2022
### Model ALL (void field) or specific modelid: 1:LASHCO, 2:UCYCO, 3:COCO
+ curl -k -s --compressed -H 'Authorization: Bearer ABD4E57D-4DCE-4443-
A57F-A94D3B06A638A' -H 'Content-Type: application/json' -H 'Connection:
keep-alive' -H 'Accept: */*' -H 'User-Agent: SMAS/1.11d' -H 'Accept-
Language: en-us' -H 'Accept-Encoding: gzip, deflate' -d
'{"uid":"dzeina","sessionkey":"E167ACD8-DBAE-4F03-9249-
232F5F126AD7","modelid":2}' -X POST https://vgate.cs.ucy.ac.cy/smas/db-
fingerprint-get.php

{"status":"ok", "uid":"dzeina",
"rows":[{"flid":1,"uid":"dzeina","time":1656767217,"timestr":"Jul 02, 2022
16:06:57","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695156941579,"y":11.911892071366,"deck":
2,"modelid":2,"foid":1,"flid:1":1,"oid":100,"height":195.46588134766,"width
":444.95190429688},{"flid":1,"uid":"dzeina","time":1656767217,"timestr":"Ju
l 02, 2022 16:06:57","buid":"vessel_2a2cf77c-91e0-41e2-971b-
e80f5570d616_1635154314048","x":57.695156941579,"y":11.911892071366,"deck":
2,"modelid":2,"foid":2,"flid:1":1,"oid":113,"height":192.55709838867,"width
":593.65753173828},{"flid":1,"uid":"dzeina","time":1656767217,"timestr":"Ju
l 02, 2022 16:06:57","buid":"vessel_2a2cf77c-91e0-41e2-971b-
```

e80f5570d616_1635154314048","x":57.695156941579,"y":11.911892071366,"deck":2,"modelid":2,"foid":3,"flid:1":1,**"oid":124,"height":126.88323974609,"width":41.296272277832**},{"flid":1,"uid":"dzeina","time":1656767217,"timestr":"Jul 02, 2022 16:06:57","buid":"vessel_2a2cf77c-91e0-41e2-971b-e80f5570d616_1635154314048","x":57.695156941579,"y":11.911892071366,"deck":2,"modelid":2,"foid":4,"flid:1":1,**"oid":104,"height":253.95126342773,"width":92.385215759277**}, ….

5 SMAS Indoor Modeling Layer and SMAS Training Layer

Main author of the chapter: Demetris Zeinalipour, UCY

For completeness, in this section we augment our description with a summary of the Training and Indoor Modeling layers that have been a primary focus of D06.4 - Background and testing of smart alert system of nearby responders. We also augment the description with necessary details that were not so important in the earlier deliverable to make our description more self-contained and complete.

5.1 Indoor Modeling Layer

Unlike outdoor environments, indoor spaces are characterized by complex topologies and are composed of entities that are unique to indoor settings, such as multiple decks, rooms and hallways connected by doors, walls, stairs, escalators, and elevators. To make things worse, doors may be one-directional (e.g., in security control doors), while temporal variations may occur (e.g., an area might be temporarily inaccessible due to its opening hours). In this section we explain how we map a vessel in the scope of the SMAS system using our in-house Anyplace service that required to undergo specific changes to support vessel environments.

5.1.1 Background on the Anyplace Architect

The Anyplace *Architect* is a web application that offers a feature-rich, user-friendly and account-based interface for managing indoor models in Anyplace. Particularly, it can be used to log-in with a Google account and place the blueprint of a building on top of Google Maps with multi-floor support. Using the floor editor, the user can upload, scale and rotate the desired blueprints to fit them properly. The user can later add, annotate and geo-tag POIs inside the building and connect them to indicate feasible paths for enabling the delivery of navigation directions. This interaction is carried out with drag-n-drop functionality that is cross- browser compatible and even operational on tablets and smartphones used in field deployments (e.g., while moving around with a tablet and correcting the indoor model).

The Architect also provides a range of other functionality, namely: i) *monitoring crowdsourcing progress* to collect fingerprints using color heat-maps. An assigner can easily identify whether a given collection is satisfactory or not and thus define quantitative acceptance criteria for the output of crowd- sourcers; ii) *making a building public or private*, which automatically shares a building on the Anyplace *Viewer* interface (given that there are no collisions). Alternatively, a building can remain private and be shared among users through a URL (e.g., a person mapping a building for a specific event publicizes a private building to its audience by email or social media); and iii) *export and import*

of indoor models and Radiomaps, which allows somebody to backup/restore a building, expedite user input of POIs, but also create a new model for a different purpose (i.e., template-based generation of a new building id for a new purpose).

5.1.2 Modeling Vessels in Anyplace

As part of Work Package 5 (Ship Integration), three generic ships were selected, based on the arrangement of ro-ro cargo spaces and passenger and cargo capacity in comparison to statistical data and trends of the world fleet, representing the three ship types: ro-ro passenger ship (Stena Flavia), ro-ro cargo ship (Magnolia Seaways, DFDS) and vehicle carrier ship (Torrens, Wallenius & Wilhelmsen). Given the relevance of both this deliverable and deliverable D06.6, we focus on the Stena Flavia vessel, but our discussion can easily be extended to any other vessel for which we have deck maps.

In the below sections we present a visual illustration of how the Stena Flavia was mapped on Anyplace Architect (along with notes on the necessity of each step). Generally, these steps are part of the localization installation and can be carried out in a few hours (e.g., 30 minutes per floor) by non-technical personnel.

The interactive Stena Flavia indoor model and vessel search engine is available at the following URL: https://anyplace.cs.ucy.ac.cy/viewer/?cuid=lashfire

Figure 13: *The Stena Flavia mapped on Anyplace Architect. This is a one-off process per vessel that takes approximately 30 minutes per deck. Modeling Vessel Interior spaces (corridors and POIs) allows us building a semantic indoor space used for search, localization, and navigation.*

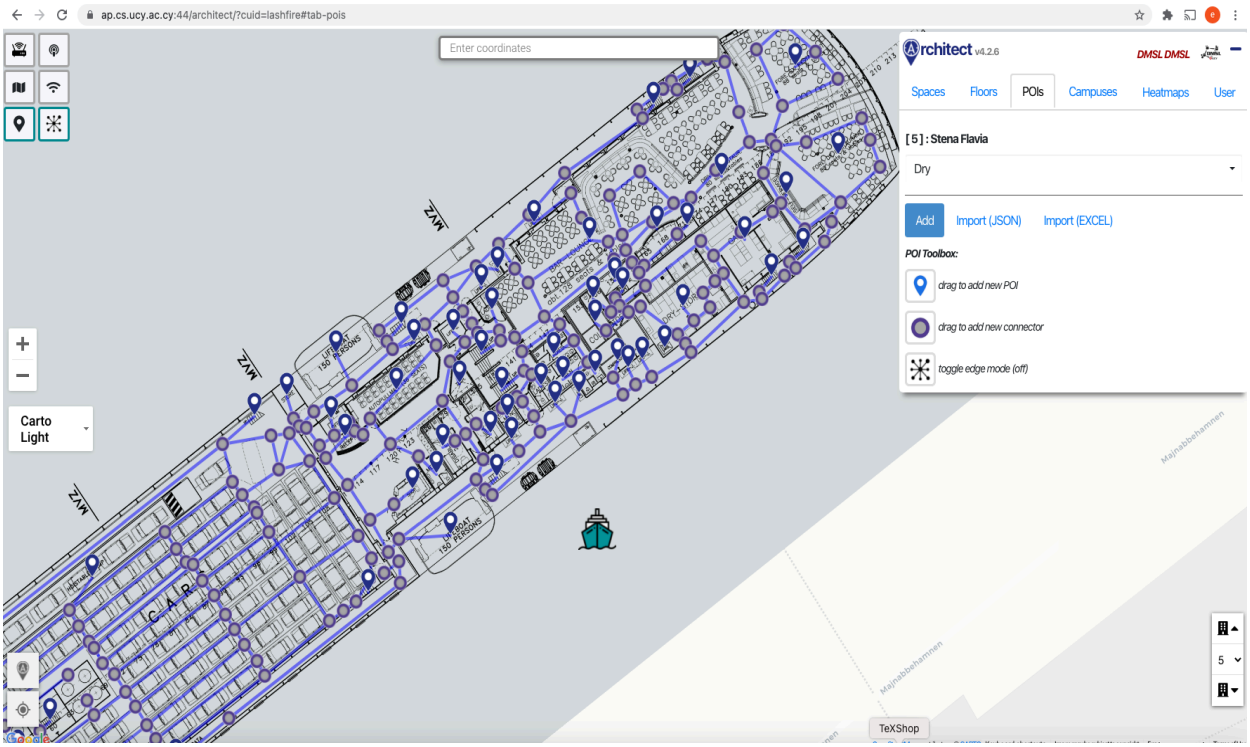


Figure 14: *Modeling Vessel Interior spaces (corridors and POIs) allows us building a semantic indoor space used for search, localization, and navigation. The above indoor model is loaded on the localization engine to provide the map, search, localization, and navigation along with chat and heat scanning.*

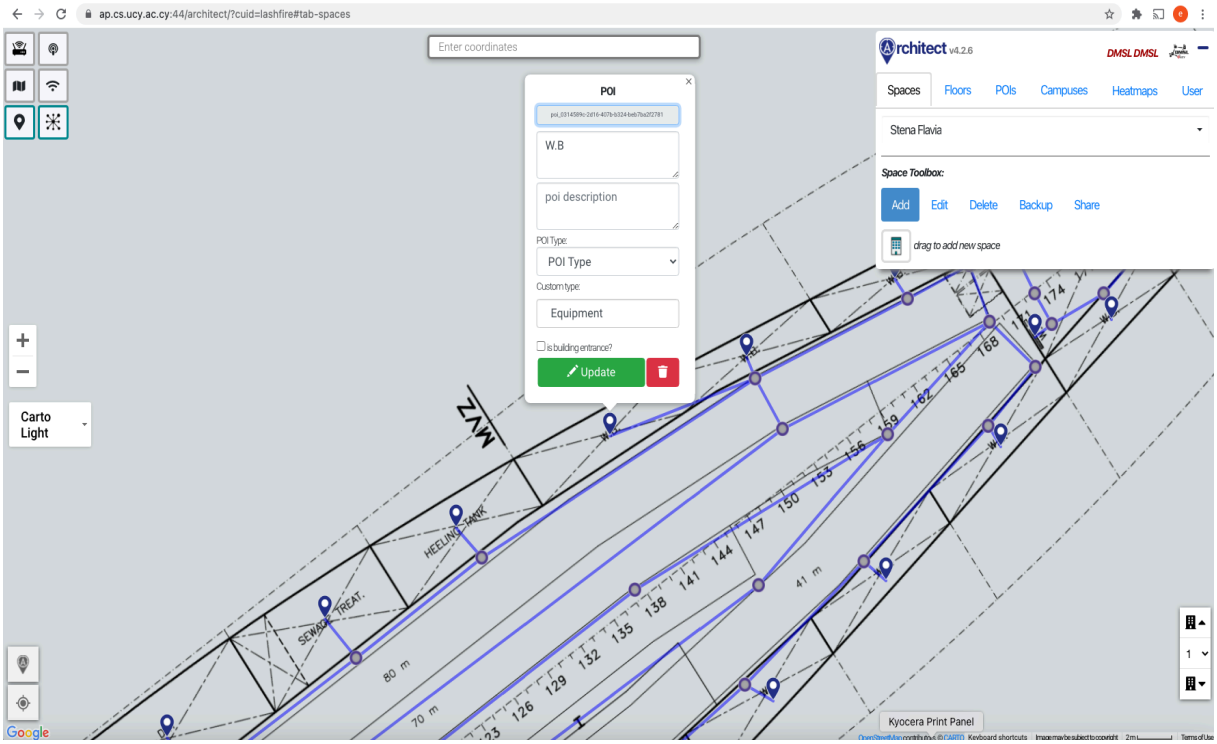


Figure 15: *POIs can be edited to customize the information space and this information will be available to first responders over their smartphone.*

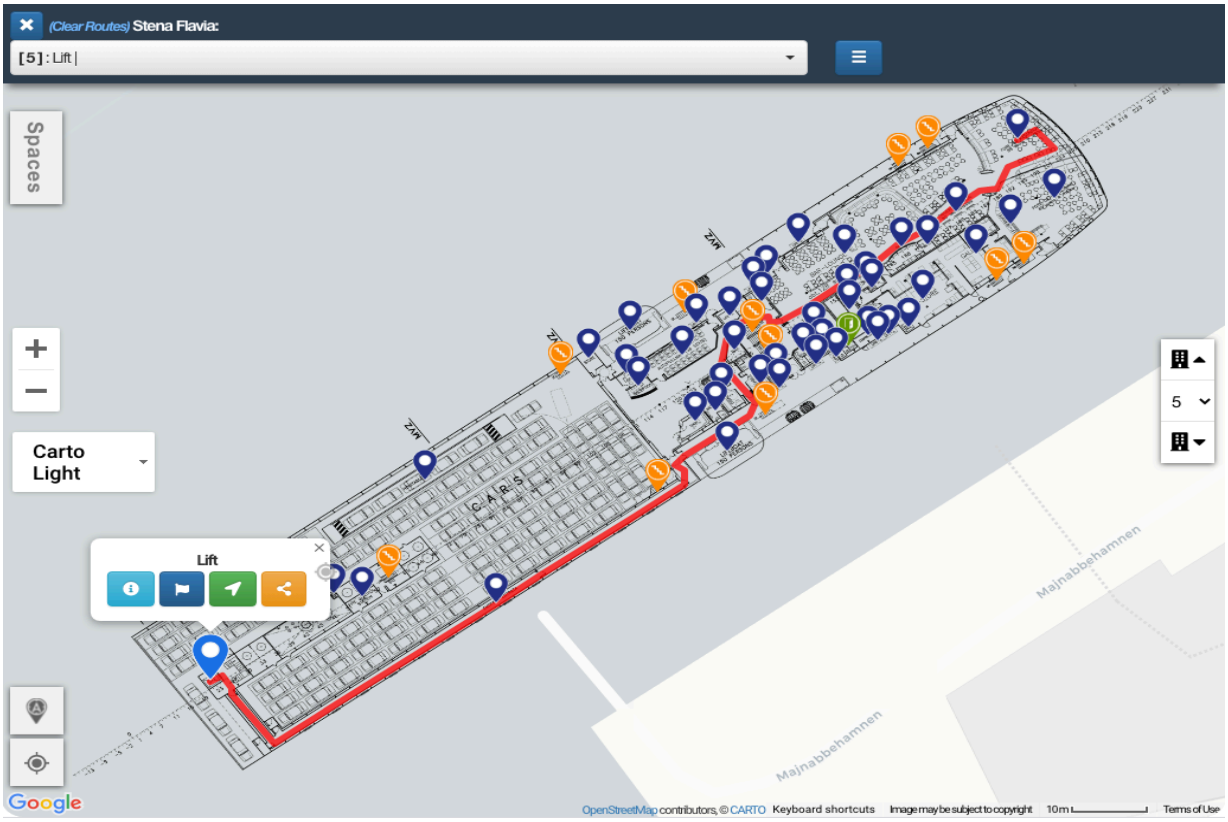


Figure 16: *The Anyplace Viewer is a search engine for the indoor model constructed in Architect. It has been adapted for vessels providing cross-deck search and navigation over a web browser or mobile application. This navigation map can also be useful to other crew members as it provides a handy all-in-one-map that can either be used*

As part of D06.6 we show how the SMAS client integrates to the Anyplace mapping service using some designated user interfaces shown in ***Figure 17***.

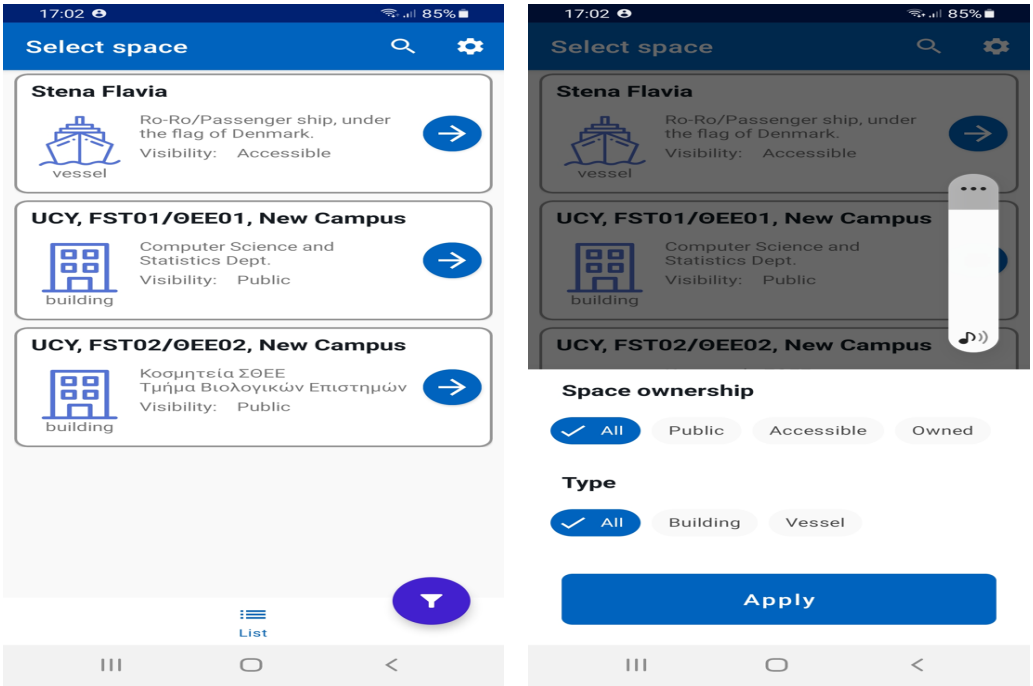


Figure 17: *Showing the Anyplace integration interfaces of SMAS, namely the building or vessel selector.*

5.2 Training Layer

Machine Learning (ML) refers to the use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data. ML training is a time-consuming process that aims to feed the so-called Neural Networks with enough feedback necessary so that a computer algorithm (i.e., a machine learning algorithm) can build a machine learning model. Using this model, the computer will be able to recognize objects with high confidence. The first important part is that we, as trainers, need to define classes (i.e., objects) of interest. The second part is that we must feed the algorithm with enough data (i.e., video traces) so that the algorithm builds the model that can predict in the future the existence of an object regardless of distractions (e.g., whether or not it appears at the same place with or without other objects, lighting conditions).

In this section, we describe how the SMAS system can recognize the interior surfaces of a vessel using state-of-the-art development in the sphere of autonomous driving cars.

5.2.1 Machine Learning Training

The problem: The simplest approach is to analyze a video frame-by-frame manually and annotate it with objects it contains. For a video with 15 fps (15 frames-per-second) this would result in 15 different annotations (i.e., 900 annotations for 1 minute of video!) and that is clearly time-consuming and cumbersome to achieve.

The solution: Fortunately, Intel developed a complete studio, coined CVAT, upon which training of machine learning models becomes practical. Computer Vision Annotation Tool (CVAT) - CVAT.org - is a free, open source, web-based image and video annotation tool which is used for labeling data for computer vision algorithms. Originally developed by Intel, CVAT is designed for use by a professional data annotation team, with a user interface optimized for computer vision annotation tasks as shown in ***Figure 18***.

5.2.2 Training Stena Flavia Classes

In this section we describe an extensive annotation of videos we carried out with the CVAT video on our dedicated deep learning datacenter. Computer Vision Annotation Tool (CVAT) - CVAT.org - is a free, open source, web-based image and video annotation tool which is used for labeling data for computer vision algorithms. Originally developed by Intel, CVAT is designed for use by a professional data

annotation team, with a user interface optimized for computer vision annotation tasks. For example, one can lock an object and have the user interface automatically track the object across several frames to minimize the burden on the trainer. Additionally, one can easily create new classes and revisit prior annotations to train for say a new class that is desired by the deployment team.

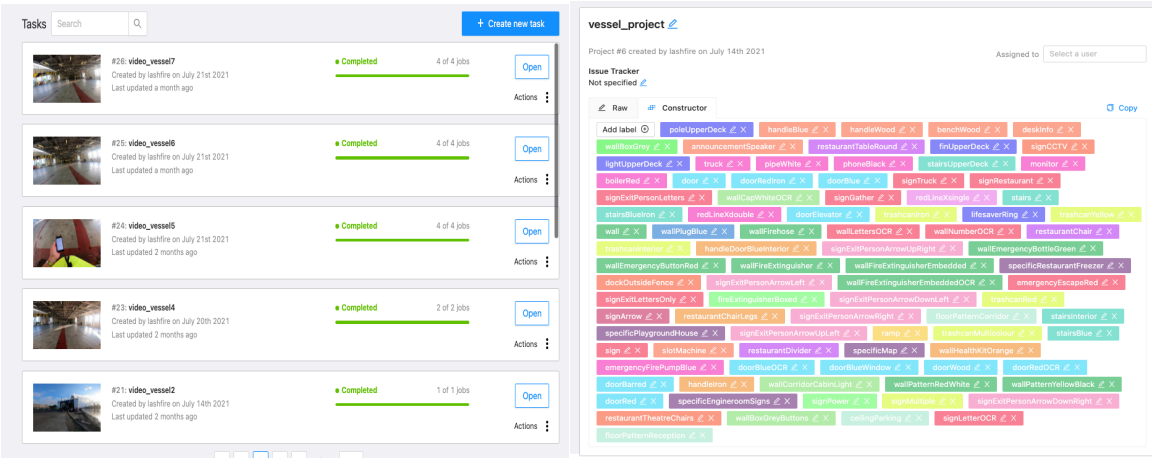


Figure 18: *(left) Intel CVAT Training Environment and Job Monitoring User Interface. (right) we show the various classes used for annotating the videos with useful semantics that are integrated in the machine learning models.*

Particularly, we obtained 15 videos from the Stena Flavia (captured by Martin Carlsson (STL) with a chest-mounted GoPro camera – not a smartphone). The videos were annotated by a team of 5 persons over a period of 4 weeks with several meetings iterations, to refine the quality of the constructed LASHCO neural network model we built for the Stena Flavia vessel. The above gave us a model that can recognize the following type of objects (i.e., classes):

- White: signs
- Yellow: doors
- Green: objects in the PAX public areas (e.g., restaurant)
- Red: walls, floors, ceilings, and objects attached to these surfaces
- Gray: stairs
- Blue: other objects.

Table 14: Labels used to Train the Stena Flavia videos and generate a respective LASHCO model.

sign	deskInfo	doorRed	specificPlaygroundHouse
signTruck	slotMachine	wallFireExtinguisherEmbedded	restaurantTheatreChairs
signGather	benchWood	wallFireExtinguisherEmbeddedOCR	handleIron
signExitPersonLetters	announcementSpeaker	emergencyEscapeRedSign	wallCorridorCabinLight

wallLettersOCR	lifesaverRing	fireExtinguisherBoxed	specificMap
wallNumberOCR	poleUpperDeck	redLineXsingle	wallHealthKitOrange
signRestaurant	finUpperDeck	doorElevator	floorPatternCorridor
stairs	lightUpperDeck	ramp	signExitLettersOnly
stairsBlue	truck	signCCTV	signExitPersonArrowDownLeft
stairsBlueIron	redLineXdouble	specificEngineroomSigns	signExitPersonArrowDownRight
stairsUpperDeck	pipeWhite	wallPlugBlue	signExitPersonArrowUpLeft
trashcanRed	phoneBlack	signPower	signExitPersonArrowUpRight
trashcanMulticolour	monitor	wallBoxGrey	signExitPersonArrowLeft
trashcanIron	boilerRed	signMultiple	signExitPersonArrowRight
trashcanYellow	emergencyFirePumpBlue	wallCapWhiteOCR	signArrow
wall	door	wallBoxGreyButtons	signNumberOCR
wallFirehose	doorRedIron	ceilingParking	specificTankDeck
wallFireExtinguisher	doorBlue	floorPatternReception	wallSingleDigitOCR
wallEmergencyButtonRed	doorBlueOCR	signLetterOCR	signLifeSaver
wallEmergencyBottleGreen	doorBlueWindow	trashcanInterior	signFireSafetyLever
restaurantDivider	doorWood	handleDoorBlueInterior	signFireGlassBreak

restaurantTableRound	doorRedOCR	stairsInterior	signFireExtinguisher
restaurantChair	doorBarred	restaurantChairLegs	signBathroomDisabled
handleWood	wallPatternRedWhite	specificRestaurantFreezer	symbolNoSmoking
handleBlue	wallPatternYellowBlack	deckOutsideFence	

Note: The above object coloring scheme is mainly for better presenting the generated classes of the model and have no relation to the colors that show up in the SMAS app when recognizing objects (there the colors are sequential and at random).

Training Equipment: To execute CVAT in our environment we have setup a dedicated rack able server featuring an NVIDIA Tesla V100 GPU card that we had to acquire and setup in our datacenter. This card reduces training time down to a few hours from several days or weeks we required initially on Google's free Colab environment (https://colab.research.google.com/). Additionally, by processing our data locally allows us to improve I/O performance (as handling large video traces over a slow network can become the bottleneck.)

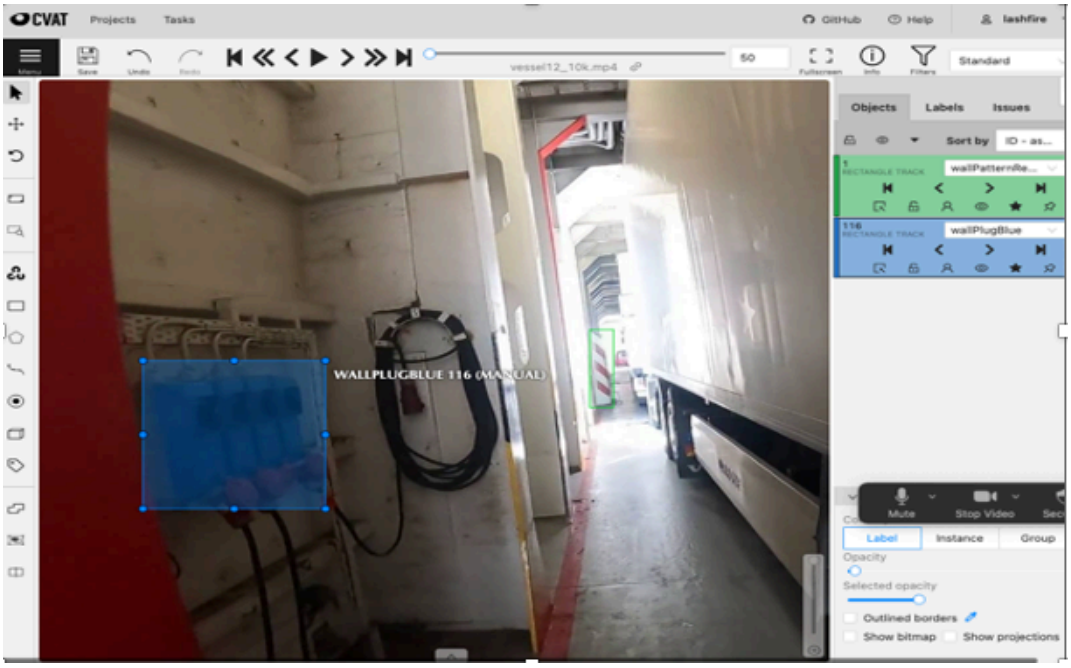


Figure 19: *Example of annotating charging stations inside the CVAT environment (we finally opted to annotate such charger clusters individually rather as one object). Revisiting the annotation and refining the model is straightforward (e.g., the revised model can be shipped as part of the next software update).*

Constructed Models:

Our SMAS architecture currently allows electing between the following three machine learning corpuses:

- **Common Objects in Context (COCO):** this is a large-scale object detection, segmentation, and captioning dataset (24MB). COCO has several features, namely: Object segmentation, Recognition in context, Superpixel stuff segmentation, 330K images (200K labeled), 1.5 million object instances, 80 object categories, 91 stuff categories, 5 captions per image, 250,000 people with keypoints. COCO was used for development and for testing the platform for localization accuracy in ordinary spaces.
- **University of Cyprus Objects in Context (UCYCO):** this is a small-scale object detection, segmentation, and captioning dataset specifically for the University of Cyprus buildings (23MB). This model was particularly useful for the development stage and was constructed with the assistance of Google Colab (i.e., GPU resources in the cloud).
- **LASH Objects in Context (LASHCO):** this is a mid-scale object detection, segmentation, and captioning dataset specifically for a vessel as part of LASH FIRE project (24MB). LASHCO has approximately 100 classes. The given object detection, segmentation, and captioning dataset will be used for testing the platform and localization accuracy in vessel spaces as part of the LASH FIRE project and has been constructed on a dedicated deep learning server, namely an HP DL380 Gen10 with 80 logical processors and a powerful NVIDIA v100 card.

Our three models cover both the Stena Flavia remote/in-person study through the LASHCO model but also our laboratory/testing environment at the University of Cyprus with the UCYCO model and the COCO model that works pretty much everywhere. LASHCO and UCYCO have been trained by our team while COCO is readily available on the Internet.

5.2.3 Managing and Optimizing Models

Exporting Models: Once the training task completes, we are able to export a YOLO (YOLO: Real-Time Object Detection) directly off the CVAT system and observe the performance of the object recognition on the laboratory machines. One problem is that the YOLO models are not sufficiently efficient on smartphones, as this was elaborated in D06.4 - Background and testing of smart alert system of nearby responders, as such we export the models in Tensorflow Lite, which is a mobile library for deploying models on mobile, microcontrollers and other edge devices. ***Figure 20*** shows the file system structure of the three models (all of which account to the very reasonable size of 72MB in total). Looking into the individual files we observe that obj.names contains the classes names in plain text while

model.tflite contains the model in tensorflow format (https://www.tensorflow.org/lite/models/convert/metadata)

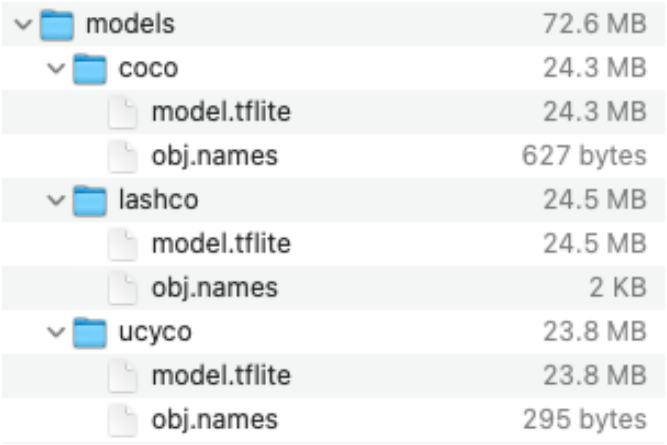


Figure 20: *Example of the exported Machine Learning Models in Tensorflow Lite format*

16-bit Quantization on Models: To reduce the size of the trained Neural Networks (NN), we chose to apply the quantization method during training. This has to do with the number of bits used to represent the floating point numeric in the NN models. Under normal circumstances, all FPN (Floating Point Number) computations are carried out on 32-bit byte sequences. With quantization however to 16-bit, we managed to dramatically reduce the NN model size by over 50%. This size reduction had a dramatic impact on the performance of the model with no great loss in accuracy. Actually, we observed only very little degradation in object recognition performance so kept 16-bit quantization default in the SMAS project. More optimizations will be considered in the future.

6 The SMAS Data Layer

Main author of the chapter: Demetris Zeinalipour, UCY

6.1 Database

6.1.1 Introduction

The SMAS Database stores all information relevant to the application layer of the application and is implemented in open-source SQLite, which is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is an embedded relational database, as opposed to a database running in an isolated process, which allows extreme flexibility with its deployment. In fact, SQLite is the most deployed database in the world as it is embedded in smartphone operating systems SDKs (e.g., iOS and Android), web browsers (Chrome, Safari), Web application frameworks (e.g., drupal, Django) and others.

One particular benefit of implementing SMAS in SQLite is that we can run the system on both a very powerful server, through a sharded database architecture we currently deploy in the SMAS cloud layer, to a low-end server running on the edge (e.g., an inexpensive Raspberry server on a small SMAS installation).

6.1.2 Database Design

The relational schema of the SMAS database is provided in ***Figure 21***.

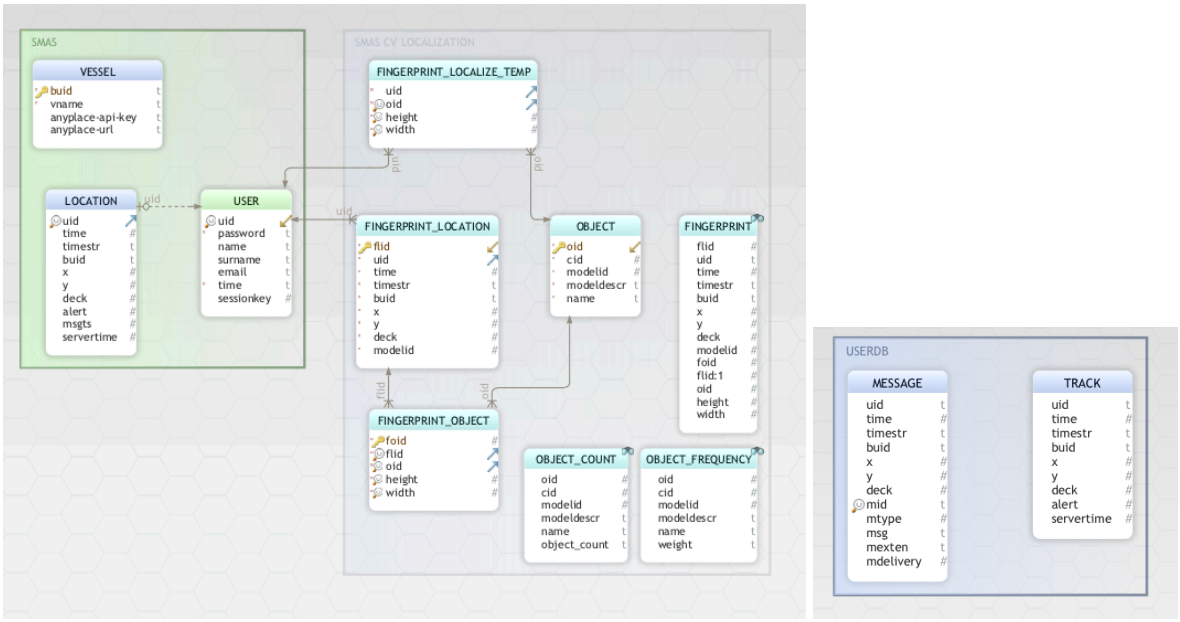


Figure 21: *SMAS Database Relational Schema (showing base tables and views) for lash.db (main database) and the sharded <user>.db of each user that allows coping with data ingestions without scalability issues.*

The diagram shows both the base tables and the views (i.e., virtual tables), which constitute the data architecture of our system. All functionality of the system is implemented in SQL-DML (Data Manipulation Language) and SQL-DDL (Data Definition Language), allowing managing in a declarative way the information space of the SMAS ecosystem.

The reason for breaking the user database down to a new micro-database is to allow faster and more efficient ingestion of data. Particularly, edge ingestion systems might be characterized by velocity of location updates and with such an architecture we provide an extremely scalable data management scenario that can scale to thousands of users without data ingestion bottlenecks and without proportionally scaling the database architecture.

In the subsequent sub-sections, we will provide an explanation of our tables and the philosophy behind our data layer.

6.1.3 User

This table contains various details the SMAS system retains per user as well as the constructed session-keys. The table is updated on each login with a newly constructed session-key of a specific user. Linking the system with an LDAP/AD directory service can be part of a future development, shall the system be integrated in a specific vendor-specific software ecosystem. ***Figure 22*** shows an instance of the table.

Table: USER

Filter in any column

	uid	password	name	surname	email	time	sessionkey
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	smas	68a07a0d-93e0-4107-a2b3-d24c7762aa31	SMAS	BATCH SYSTEM	smas@cs.ucy.ac.cy	2022-02-18 07:42:01	
2	pmpeis	1234	Paschalis	Mpeis	pmpeis@ucy.ac.cy	2022-02-18 07:43:10	8AFE7054-1FE9-491B-96C1-110EFE808A4D
3	maria	1234	Maria	Hjohlman		2022-02-18 07:43:23	FB4E2003-9B9A-4F3A-9D5D-E4F88B95D7D4
4	jaime	1234	Jaime	Bleye Vicario	jaimebv@centrojovellanos.es	2022-02-18 07:43:52	4F7703AB-3A18-4526-9CDB-93E0C42BD0F7
5	martin	1234	Martin	Carlsson		2022-02-18 07:44:07	5ABF8523-E8AC-4CFE-A4E7-A773A09D8F9C
6	dzeina	1234	Demetris	Zeinalipour	dzeina@ucy.ac.cy	2022-02-18 07:42:01	582F5FD7-16CF-4F28-A067-078649A9C199
7	vito	1234	Vito	Radolović	NULL	2022-02-22 17:02:49	61ACE202-32DF-4E3F-839C-77AE700F8738
8	test	1234	Test	Test		2022-02-23 10:10:12	A14B8B45-8B86-48A9-9EF3-58BFE9B57BB9
9	paschalis	1234	Paschalis	Mpeis	pmpeis@ucy.ac.cy	2022-03-22 14:23:56	FE6A8E1A-27F9-49CC-A637-4957F2FF653A
10	athina	1234	Athina	Hadjichristodoulou	NULL	2022-03-22 14:42:43	8EFD6143-F277-46CD-B50B-02AD943571A7
11	panagiotis	1234	Panagiotis	Vassiliou		2022-03-22 14:47:51	NULL

Figure 22: *SMAS Database – Example User Table*

6.1.4 Location

This table contains the latest location status of each user in the system. This information is consolidated from a per-user database that contains the TRACK and MESSAGE tables, each containing the location trajectory of a user and the messages on the chat wall of each user. This table is re-

constructed automatically by a system cronjob so that it continuously contains the latest location of users. ***Figure 23*** shows an instance of the table.

Table: LOCATION — Filter in any column

	uid	time	timestr	buid	x	y	deck	alert	msgts	servertime
1	athina	1656949771	Jul 04,2022 18:49:31	vessel_2a2cf77c-91e0-41e2-971b-...	57.69515142542	11.912214716228	5	0	1656949256	2022-07-04 15:49:34
2	dzeina	1656949261	Jul 04,2022 18:41:01	vessel_2a2cf77c-91e0-41e2-971b-...	57.69513418577	11.91200774163	2	0	1656949256	2022-07-04 15:49:34
3	jaime	1656949771	Jul 04,2022 18:49:31	vessel_2a2cf77c-91e0-41e2-971b-...	57.695278672542	11.912824462278	1	0	1656949256	2022-07-04 15:49:34
4	maria	1656949771	Jul 04,2022 18:49:31	vessel_2a2cf77c-91e0-41e2-971b-...	57.695187742542	11.912184126228	5	0	1656949256	2022-07-04 15:49:34
5	martin	1656949770	Jul 04,2022 18:49:30	vessel_2a2cf77c-91e0-41e2-971b-...	57.695266222542	11.912278962278	1	0	1656949256	2022-07-04 15:49:35
6	paschalis	1656430627	Jun 28,2022 18:37:07	vessel_2a2cf77c-91e0-41e2-971b-...	57.695249039745	11.912561282516	3	0	1656949256	2022-07-04 15:49:35
7	pmpeis	1656944606	Jul 04,2022 17:23:26	vessel_2a2cf77c-91e0-41e2-971b-...	57.695147086703	11.911815293133	1	0	1656949256	2022-07-04 15:49:35
8	vito	1656949771	Jul 04,2022 18:49:31	vessel_2a2cf77c-91e0-41e2-971b-...	57.695244422542	11.912191026228	1	0	1656949256	2022-07-04 15:49:35

Figure 23: *SMAS Database – Example Location Table*

Consolidating the data from the sharded databases in a single global database minimizes exclusive write locks on the central lash.db database. Given that all endpoint calls are initiated from the web, i.e., PHP calls, we use exclusive and shared locks (i.e., semaphores) to manage race conditions and have the SMAS service operational even at extremely high stress scenarios.

6.1.5 Fingerprint (Database)

This view contains the objects registered through the SMAS Logger by user. Particularly, it records for various (vesselid, x, y, deck)-quadruples the oids (object-ids) and dimensions of the oids identified at the given location. This view plays a central part in the localization query as this is the table that our proposed Surface algorithm exploits to find the highest matched (vesselid, x, y, deck)-quadruples.

Table: FINGERPRINT — Filter in any column

	fiid	uid	time	timestr	buid	x	y	deck	modelid	foid	fiid:1	oid	height	width
1	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	1	1	100	195.46588134766	444.95190429688
2	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	2	1	113	192.55709838867	593.65753173828
3	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	3	1	124	126.88323974609	41.296272277832
4	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	4	1	104	253.95126342773	92.385215759277
5	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	5	1	117	89.259399414062	75.437004089355
6	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	6	1	104	203.76571655273	108.2600402832
7	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	7	1	104	277.25186157227	155.26919555664
8	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	8	1	104	217.48486328125	121.38284301758
9	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	9	1	112	59.647033691406	384.26556396484
10	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	10	1	104	202.58538818359	115.83800506592
11	1	dzeina	1656767217	Jul 02,2022 16:06:57	vessel_2a2cf77c-91e0-41e2-971b-...	57.695156941579	11.911892071366	2	2	11	1	113	54.536163330078	357.20443725586
12	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	12	2	113	70.413665771484	400.20138549805
13	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	13	2	113	59.740570068359	444.47717285156
14	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	14	2	113	67.466827392578	399.4931640625
15	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	15	2	113	56.131958007812	435.63589477539
16	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	16	2	124	52.464874267578	56.923583984375
17	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	17	2	124	65.821258544922	47.544845581055
18	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	18	2	113	68.640991210938	414.39190673828
19	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	19	2	113	54.618530273438	394.56887817383
20	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	20	2	104	193.73492431641	55.785747528076
21	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	21	2	113	63.615203857422	399.74237060547
22	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	22	2	104	182.01235961914	115.67166137695
23	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	23	2	113	64.215942382812	403.60202026367
24	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	24	2	113	51.280639648438	386.97857666016
25	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	25	2	124	54.286041259766	51.463302612305
26	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	26	2	113	45.660095214844	375.92224121094
27	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	27	2	124	52.833312988281	49.884613037109
28	2	dzeina	1656767229	Jul 02,2022 16:07:09	vessel_2a2cf77c-91e0-41e2-971b-...	57.695165004657	11.911921575665	2	2	28	2	124	44.440765380859	47.486145019531

Figure 24: *SMAS Database – Example Fingerprint (Database) Table*

The view is constructed from 2 base tables that minimize the duplication of data through a 1:M relationship on the Entity Relationship Diagram. ***Figure 24*** shows an instance of the view.

Fingerprint Normalization: The view has dynamically always the latest state of the fingerprints, as opposed to our Anyplace architecture that would traditionally carry out a normalization step on Wi-Fi fingerprints to remove possible noise due to the nature of radio signals. With the Computer Vision (CV) fingerprints however, the requirement of normalization on the fingerprints is not necessary, mainly because CV fingerprints contain in most cases no noise.

Particularly, whatever the camera system captures are what we log in the Fingerprint Database, while with Wi-Fi signals the situation is not the same. In the latter, signals can come from many hundreds of meters away, exposing low attenuation and comprising the normalization step a requirement. CV fingerprints might in certain cases wrongly classify objects (e.g., seeing a refrigerator as a desk), however we observe that these recognitions are consistent, meaning that the fingerprint consistency is not compromised. We in fact exploit this parameter in our algorithm given that we focus on object frequencies and having diversity in the sort of objects we see is a benefit.

Another reason why we don't need to normalize is that the SQL layer absorbs the I/O latency making it practical to have fingerprint database in order of millions of rows without a great impact on the localization latency (i.e., queries can be performed within milliseconds without being the bottleneck).

Nevertheless, optimizing data access further is straightforward through the provision on SQL indices, even though our base installation did not require us touching upon this aspect.

NMS Threshold in Yolo: YOLO uses Non-Maximal Suppression (NMS) to only keep the best bounding box. The first step in NMS is to remove all the predicted bounding boxes that have a detection probability that is less than a given NMS threshold. We found this parameter quite useful for the SMAS Logging part as it made fingerprints compact, given that we usually had sufficient times to carry out multiple logging iterations for various scenes. For the localization case however and especially the tracking situation, the situation is reversed. Here we want to hoover as many objects as possible in a small amount of time as the user is moving quickly. Given that the inference is carried out at 640x480, cropped 416x416 with detection time around 850 ms, implies that we might not have enough time to build confidence in the recognized objects using the NMS threshold. As such, we decided to disable the YOLO NMS filtering and we observed significant gains in the localization tasks.

6.1.6 Fingerprint (Query)

This table contains the latest localization query for each user. It particularly records the oids a person has seen, and the localization algorithm will compute the closest match according to the Surface algorithm we propose. The reason for storing the latest request in a query is to make the Surface algorithm a single SQL query (i.e., all inputs require to be relations). ***Figure 25*** shows an instance of the table.

Table: FINGERPRINT_LOCALIZE_TEMP

	uid	oid	height	width
	Filter	Filter	Filter	Filter
1	dzeina	100	135.08544921875	249.04969787598
2	dzeina	103	72.872894287109	183.72430419922
3	dzeina	111	23.2	23.2
4	dzeina	123	23.2	23.0

Figure 25: *SMAS Database – Example Fingerprint Table*

6.1.7 Object & Object Frequencies Tables

Our Machine Learning Computer Vision models are represented in the system by the means of the OBJECT table, which provides the oid for each type of object our system recognizes (i.e., the model weights are stored in the filesystem). Our localization aims to exploit the underlying frequencies of appearance of objects in the ranking process. For this reason, we construct some database views that count the number each object has been found in the FINGERPRINT view (i.e., the OBJECT_COUNT view), upon which the OBJECT_FREQUENCY table is constructed. The FREQUENCY view is central as it captures the importance an object plays in the localization process as this will be explained in Section 6.3, where we discuss our Surface algorithm.

Rare Object => High Weight in Localization

Frequent Object => Low Weight in Localization

6.2 Background on Structured Querying

The surface algorithm is a state-of-the-art localization algorithm we have developed in the Structured Query Language (SQL) to compare a fingerprint of user uid against the fingerprints stored in the database. **Structured Query Language (SQL)** is a standardized programming language that is used to manage relational databases and perform various operations on the data in them. We use this powerful programming language as a novel paradigm to implement a state-of-the-art algorithm, where localization algorithms are traditionally implemented in imperative programming languages (featuring

control loops and other programming constructs). In the subsequent paragraph, we explain the advantages of this approach:

Expressing the localization algorithm in SQL has the following benefits:

- **Declarative:** It exploits a declarative query language that has expressive power without getting into the implementation details – we describe what we want not how we want it. This allows quickly prototyping complex ideas that would have otherwise taken numerous man-hours to be developed in an imperative host language like Kotlin (i.e., the SMAS front-end layer).
- **Relational:** It allows expressing the localization task as a sequence of set-theoretic operators on a relationally-complete language (i.e., SQL), which is founded on the mathematical pillars of the relational algebra, upon which the complete relational data management field is founded.
- **Structured:** Perceiving the data in relations with relationships allows easier development of ideas as data is organized in tabular form with constraints and foreign keys.
- **Performance:** It allows testing important aspects (like query response time) early on in the algorithm development stage and applying respective remedies if needed (e.g., indices, views, etc.) Input/Output performance tuning now becomes a first-class citizen.
- **Portability:** The code runs directly both in the cloud on an extremely powerful server and on a low-end device. More importantly, it runs on the SMAS android app written in Android that is natively supported by SQLite and SQL.
- **Data-driven Algorithm Design:** We are able to design the algorithm by looking at the data through various SQL predicates. We found this extremely powerful in designing and progressing the logic of our algorithm.

6.3 The Surface CV Localization Algorithm

6.3.1 Overview

In this section we describe our Surface algorithm, explaining its rational through various examples. The Surface algorithm uses a combination of Global and Local pruning strategies, where the **Surface Global (SG) Algorithm** is responsible to find a location through all floors on first sight while the **Surface Local (SL) Algorithm** is responsible to find the next location based on the prior (x, y, deck). This is particularly useful for tracking scenarios, where a continuous location is anticipated on the map, and we don't want very radical location shifts.

The SL & SG algorithms deploy sophisticated object ranking functions founded on the following concepts and are implemented with the expressive power of SQL:

- ***Multiset Subtraction (SG & SL algorithms),*** *where* it scans through collected fingerprints and identifies the (x,y,deck) locations that resemble more closely the set of objects in the query set.
- **Global Partitioned Frequency Counting *(SG & SL algorithms)***, which captures the frequency a given object has after the SMAS logging stage. This way, the Surface algorithm knows which objects are of high importance and takes those objects into the ranking process of (x,y,deck) candidates. The objective is to return the (x,y,deck) whose object set resembles more closely the object set of the query. Clustering of the frequencies is carried out with spatial hashing.
- **Spatial Partitioning of Fingerprints *(SG & SL algorithms)*,** which clusters close-by fingerprints based on a system-derived clustering parameter (in our setting 10 meters) that allows more accurate ranking of location similarity results.
- **Bounding Box Filtering of Fingerprints *(SL algorithm)*,** which applies to the case of the local algorithm and tracking scenarios where location estimates are aimed to be at most 100 meters and +/- 1 deck apart. If the bounding box filter is too aggressive leaving no results, the global localization algorithm alleviates the problem and finds temporality the best result until a better estimate can be made.

6.3.2 Multiset Subtraction

A *set* in Mathematics is generally defined as an unordered collection of distinct objects. In the case of objects captured during CV logging, a given object might appear multiple times (e.g., an area contains multiple drenchers). As such, we relax the discussion and adopt the notion of a *multiset,* which allows the repetition of objects in the collection. A multiset difference is generally defined in the set builder notation as $A - B = \{x \mid x \in A \text{ and } x \notin B\}$, considering that the set is permitted to contain duplicates.

In Table 15 we discuss the basic idea of multiset subtraction with a few examples.

Table 15: Multiset Subtraction used in the Surface Algorithm

Example Multiset Subtraction (Query – FDB Record):
• {drencher, charger} – {drencher, charger} = Ø
Dissimilarity = 0
• {drencher, charger} – {drencher, charger, door} = Ø

Dissimilarity = 0

- {drencher, charger} – {drencher} = {charger}

Dissimilarity = 1

- {door, door, door} – {door} = {door, door}

Dissimilarity = 2

- {door} – {door, door, door} = Ø

Dissimilarity = 0

Figure 26: *Multiset Subtraction with a real CV Example. If the left figure is the query (and the right the database), then the dissimilarity is 2 (specificEngineRoomSign not found, also doorBlueOCR found only once). If the other way around, the dissimilarity is 1 (signTruck not found)*

6.3.3 Global Partitioned Frequency Counting

The Fingerprint view in the SMAS data layer contains all the (x,y,deck) locations that have been collected during the SMAS Logging function. For an object, whether unique or frequent, we might have a dozen of fingerprints depending on much effort the installer provider during the logging process. So,

the number of fingerprints is clearly not a good indication of whether an object is unique and whether this uniqueness can be exploited to yield a high accuracy localization.

To alleviate this problem, we decided to carry out a spatial hashing of objects to locations. This allows identifying how unique an object is in relation to it location (e.g., if a signGather appears in three locations then this should have frequency counter of 3, otherwise if these three locations map to the same location the frequency counter must be 1).

In ***Figure 27***a, we show examples from both the UCYCO and LASHCO logging tasks. We can observe that certain objects appear only a few times in our logging, so these objects must be exploited in the ranking process. Particularly, missing a less important object (e.g., Trash Can in UCYCO) during the query should not penalize the query dissimilarity so much against an important object (e.g., Statue). ***Figure 27***b shows the normalized **OBJECT_COUNT** that we coin the **OBJECT_FREQUENCY** weight that is also taken into direct account in the Surface localization algorithm as it divides the **OBJECT_FREQUENCY** over the total count of object appearances in each model set. The above statistics will be exploited in both the SG and the SL algorithms we will present next.

Table: OBJECT_COUNT

	oid	cid	modelid	modeldescr	name	object_count
	Fil...	Filter	Filter	Filter	Filter	Filter
1	52	51	1	lashco	wallFireExtinguisherEmbedded	9
2	69	68	1	lashco	signLetterOCR	8
3	43	42	1	lashco	doorBlue	7
4	75	74	1	lashco	deckOutsideFence	6
5	59	58	1	lashco	signCCTV	6
6	49	48	1	lashco	wallPatternRedWhite	6
7	18	17	1	lashco	wallFireExtinguisher	6
8	4	3	1	lashco	signExitPersonLetters	6
9	3	2	1	lashco	signGather	6
10	73	72	1	lashco	restaurantChairLegs	5
11	70	69	1	lashco	trashcanInterior	5
12	55	54	1	lashco	fireExtinguisherBoxed	5
13	46	45	1	lashco	doorWood	5
14	44	43	1	lashco	doorBlueOCR	5
15	30	29	1	lashco	lifesaverRing	5
16	99	98	1	lashco	symbolNoSmoking	4
17	96	95	1	lashco	signFireGlassBreak	4
18	94	93	1	lashco	signLifeSaver	4
19	88	87	1	lashco	signExitPersonArrowLeft	4
20	71	70	1	lashco	handleDoorBlueInterior	4
21	50	49	1	lashco	wallPatternYellowBlack	4
22	19	18	1	lashco	wallEmergencyButtonRed	4
23	10	9	1	lashco	stairsBlueIron	4
24	97	96	1	lashco	signFireExtinguisher	3
25	53	52	1	lashco	wallFireExtinguisherEmbeddedOCR	3
26	13	12	1	lashco	trashcanMulticolour	3
27	12	11	1	lashco	trashcanRed	3
28	5	4	1	lashco	wallLettersOCR	3
29	2	1	1	lashco	signTruck	3
30	83	82	1	lashco	signExitLettersOnly	2
31	79	78	1	lashco	wallCorridorCabinLight	2

Table: OBJECT_FREQUENCY

	oid	cid	modelid	modeldescr	name	weight
	Fil...	Fil...	Filter	Filter	Filter	Filter
1	98	97	1	lashco	signBathroomDisabled	0.00515463917525773
2	95	94	1	lashco	signFireSafetyLever	0.00515463917525773
3	93	92	1	lashco	wallSingleDigitOCR	0.00515463917525773
4	92	91	1	lashco	specificTankDeck	0.00515463917525773
5	91	90	1	lashco	signNumberOCR	0.00515463917525773
6	90	89	1	lashco	signArrow	0.00515463917525773
7	82	81	1	lashco	floorPatternCorridor	0.00515463917525773
8	76	75	1	lashco	specificPlaygroundHouse	0.00515463917525773
9	74	73	1	lashco	specificRestaurantFreezer	0.00515463917525773
10	72	71	1	lashco	stairsInterior	0.00515463917525773
11	68	67	1	lashco	floorPatternReception	0.00515463917525773
12	67	66	1	lashco	ceilingParking	0.00515463917525773
13	64	63	1	lashco	signMultiple	0.00515463917525773
14	60	59	1	lashco	specificEngineroomSigns	0.00515463917525773
15	57	56	1	lashco	doorElevator	0.00515463917525773
16	47	46	1	lashco	doorRedOCR	0.00515463917525773
17	35	34	1	lashco	redLineXdouble	0.00515463917525773
18	33	32	1	lashco	lightUpperDeck	0.00515463917525773
19	23	22	1	lashco	restaurantChair	0.00515463917525773
20	22	21	1	lashco	restaurantTableRound	0.00515463917525773
21	83	82	1	lashco	signExitLettersOnly	0.0103092783505155
22	79	78	1	lashco	wallCorridorCabinLight	0.0103092783505155
23	78	77	1	lashco	handleIron	0.0103092783505155
24	77	76	1	lashco	restaurantTheatreChairs	0.0103092783505155
25	62	61	1	lashco	signPower	0.0103092783505155
26	61	60	1	lashco	wallPlugBlue	0.0103092783505155
27	58	57	1	lashco	ramp	0.0103092783505155
28	45	44	1	lashco	doorBlueWindow	0.0103092783505155
29	32	31	1	lashco	finUpperDeck	0.0103092783505155
30	31	30	1	lashco	poleUpperDeck	0.0103092783505155
31	28	27	1	lashco	benchWood	0.0103092783505155

Figure 27: *SMAS Database – Example of OBJECT_COUNT and OBJECT_FREQUENCY Views*

6.3.4 Spatial Partitioning of Fingerprints

Even though The original OBJECT_COUNT idea clusters fingerprints on (oid,x,y,deck), one problem is that the fingerprints will rarely be precisely on the same (x,y) because Location in Google Maps is in WGS84 (that we use for the mapping layer has very high precision.)

The below shows one example of an (x,y) point we represent in the SMAS application as WGS84 location (longitude, latitude) pair: (57.695137769363, 11.911948062479). Looking at decimal precision in the WGS84 we observe that by rounding to four decimal places gives as a practical mechanism to cluster close-by fingerprints to avoid over-counting objects. Particularly, with 4 decimal places (https://wiki.openstreetmap.org/wiki/Precision_of_coordinates) we get a clustering of 10 meters, which would mean that our ranking will now take exploit this improved OBJECT_FREQUENCY in the ranking method. The above clustering will be exploited in both the SG and the SL algorithms.

Table 16: Decimal Precision of the WGS84 formatted location gives an improved method on clustering nearby Fingerprints. We particularly use 4 decimal places, i.e., an accuracy of about 11.112 meters

Accuracy	**DD.dddddd° decimal places**
10 m	**4**
1 m	5
0.1 m	6

Table 17: The OBJECT_COUNT with WGS84 Clustering Implementation in SQL

```
CREATE VIEW OBJECT_COUNT AS
SELECT O.*, COUNT(*) AS object_count
FROM (
    SELECT F.*, O.*
    FROM FINGERPRINT F, OBJECT O
    WHERE F.oid = O.oid
    GROUP BY F.oid, ROUND(F.x,4), ROUND(F.y,4), F.deck
    -- ORDER BY  O.oid ASC;
) AS A, OBJECT O
WHERE A.oid = O.oid
GROUP BY A.oid
ORDER BY object_count DESC
```

6.3.5 Bounding Rectangle Filtering of Fingerprints

One problem with the discussion so far is that we consider object ranking to be a global task, namely an object contributes the same way to a ranking whether it appears a few meters apart from the prior location or whether it is 3 decks above. To this end, we endanger resolving objects sets to irrelevant locations on the vessel in tracking scenarios (i.e., where the localization task takes place every few seconds or after the motion sensor triggers a re-computation).

In order to alleviate this problem, we aim to rank object similarities based on the prior location of a user (prevX, prevY, prevDeck). To do so, we create a bounding rectangle around the prior location and carry out a counting and ranking of objects in that area only. This allows finding the closest most relevant object set to the query, which effectively will not be very far from the prior location. In our case, we set the bounding rectangle to be 100 meters and not more than +/- 1 floor, but these parameters are configurable. In case this threshold does not yield any relevant fingerprint (e.g., a user entering an elevator and coming out a few floors away), we automatically run the SG algorithm that will for sure find the closest match through the complete vessel (if such a match exists). The above idea has been incorporated in the SL algorithm with superb utility as our experimentation will show.

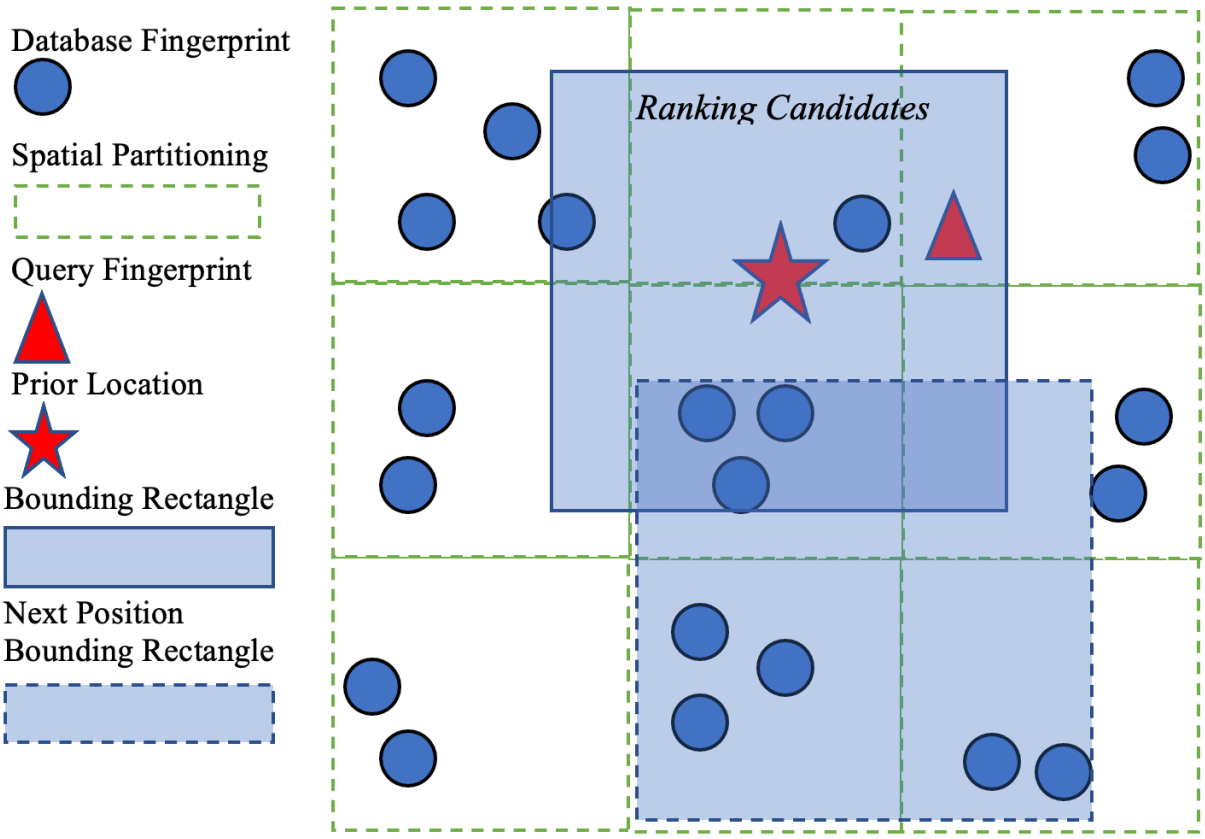


Figure 28: *Example with Bounding Rectangle Filtering of Fingerprints in the Surface Local (SL) Algorithm*

In order to visualize the Bounding Rectangle, consider the example in ***Figure 28***. Here we show the fingerprint database clustered using the Spatial Partitioning we introduced earlier. The sample also shows how the bounding rectangle drawn around the prior location reduces the prospective database fingerprints that will be examined for relevance to the query fingerprint. We observe that the prospective objects and the query object are now in spatial locality, as opposed to having the complete fingerprint database as the search object space. This means that the next location will be derived from the bounding rectangle and not from the complete spatial database, which makes the localization more accurate.

6.3.6 The Surface Global (SG) Algorithms

The Surface algorithm is split into two parts, a global algorithm that finds the location of a first responder across all decks of a vessel and the local algorithm that finds the closest location of a first responder relative to the prior location. Having covered in detail the underlying design principles of the Surface algorithms, we are now ready to present the two algorithms in greater detail.

In Table 18 we present the implementation of the **Surface Global (SG) Algorithm** in SQLite. The given algorithm runs on both the cloud/edge layer (for online localization) and the device layer (for offline localization). This algorithm will run if no prior location is known or in case the **Surface Local (SL) Algorithm**, we present next, returns an empty result (i.e., is not able to resolve the location of a user). We can observe that the basic operation is founded on the idea of a multiset subtraction that derives the **dissimilarity** of a fingerprint query to the fingerprint database. It does so by exploiting the results of a **Global Partitioned Frequency Counting** results that indicate how rare a certain object is in the fingerprinting effort.

In order to consolidate close-by fingerprints it carries out a **Spatial Partitioning of Fingerprints,** which rounds fingerprints based on $smas_db_location_bound_rounding digits of WGS84 decimal precision. In our setting, this parameter is set to 4 yielding a clustering of around 10 meters. Finally, in order to avoid return irrelevant results to the user we only return fingerprint database results that have at least one common object similarity to the fingerprint query. (i.e., if a database fingerprint F contains no overlap to the query then F will never be a candidate result).

The clause IFNULL(AVG(weight),1) penalizes objects we have not observed in the fingerprinting process as it provides them a high weight (i.e., the end of the result returned to the user). Even though we might have many candidate locations as an end result, we limit the results to 1 as a user aiming to localize can only take 1 position as its final location. Returning more than one results can be drawn by

removing the LIMIT 1 filter, however then the client needs to decide what result to choose. We think that localization must be seamless, the service carries out complex ranking ideas and returns precisely one result to the user.

Table 18: The Surface Global (SG) Algorithm Implementation in SQL

Surface Global (SG) Algorithm in SQL (SQLite)

```
   SELECT F.flid, F.X, F.Y, F.deck, (SELECT IFNULL(COUNT(*),0) FROM
        ( -- simulate except all operator with rowid
        SELECT ROW_NUMBER() OVER (PARTITION BY FLT.oid) AS RowNum, FLT.oid FROM
FINGERPRINT_LOCALIZE_TEMP FLT WHERE FLT.uid="$uid"
        EXCEPT
        SELECT ROW_NUMBER() OVER (PARTITION BY FO.oid) AS RowNum, FO.oid FROM
FINGERPRINT_OBJECT FO WHERE FO.flid=F.flid
        )
    ) AS dissimilarity, (SELECT IFNULL(AVG(weight),1) FROM
        ( -- simulate except all operator with rowid
        SELECT ROW_NUMBER() OVER (PARTITION BY FLT.oid) AS RowNum, FLT.oid,
OF.weight as weight FROM FINGERPRINT_LOCALIZE_TEMP FLT, OBJECT_FREQUENCY OF WHERE
FLT.oid =  OF.oid and FLT.uid="$uid"
        EXCEPT
        SELECT ROW_NUMBER() OVER (PARTITION BY FO.oid) AS RowNum, FO.oid, OF.weight
as weight FROM FINGERPRINT_OBJECT FO,  OBJECT_FREQUENCY OF WHERE   FO.oid =  OF.oid
and FO.flid=F.flid
        )
    ) AS weight
    FROM FINGERPRINT F
    WHERE F.modelid="$modelid" and F.buid="$buid"
    GROUP BY ROUND(F.X,"$smas_db_location_bound_rounding"),
ROUND(F.Y,"$smas_db_location_bound_rounding"), F.deck
    HAVING dissimilarity <
(SELECT COUNT(*) from FINGERPRINT_LOCALIZE_TEMP FLT WHERE  FLT.uid="$uid")
    ORDER BY dissimilarity, weight ASC
    LIMIT 1; -- sort by newest match to oldest MATCH
```

In the future we aim to deploy more complex spatial operators by deploying a designated extension (e.g., SpatiaLite). This will provide more expressive power in capturing the importance of further optimization criterions, e.g.., closeness of objects based on rectangle size, orientation, and map-matching for grounding localization requests to the underlying indoor graph topology we maintain but also more study with capturing in low-light conditions (e.g., first responder with torch for which our preliminary findings suggest that we can retain good localization accuracy).

6.3.7 The Surface Local (SL) Algorithm

The second algorithm, coined the **Surface Local Algorithm (SL),** is the default algorithm that will run in most cases of continuous localization (i.e., tracking). As such, this algorithm is tuned on the same philosophy with the global SG algorithm, namely ***Multiset Subtractions,*** **Global Partitioned Frequency Counting and Spatial Partitioning of Fingerprints**, but with spatial clustering refinements that will allow this algorithm to remain accurate when progressing from a prior known location to a new close by location. This is very often the case, also in the SMAS system, where we want to know our location

every X seconds (i.e., tracking). In Table 19 we present the SL algorithm implemented in SQL (for SQLite). The main difference of the algorithm is observed towards the end where we carry out a **Bounding Rectangle Filtering** and grouping of results based on the prior known location in order to ensure that the location estimate is not more than **$smas_db_location_bound_meters away** (in our case 100 meters and not more than 1 floor). Of course, if the SL algorithm yields no result (e.g., a first responder takes the elevator and moves 3 decks down), the SG algorithm engages automatically and will localize the first responder as soon as objects are recognized in the whatever deck the first responder ends up. Even though our algorithm has a filtering radius of about 100 meters and +/- one floor, the spatial **Spatial Partitioning of Fingerprints** is again on the granularity of 10 meters in order to tackle the fact that close-by fingerprints need to be grouped to increase the localization confidence.

Table 19: The Surface Local (SL) Algorithm Implementation in SQL

Surface Local Algorithm (SLA) in SQL (SQLite)

```
     SELECT F.flid, F.X, F.Y, F.deck,
                      ABS(x - "$prevX") as xDiff, ABS(y - "$prevY") as yDiff,
ABS(deck - "$prevDeck") as deckDiff,
                      (SELECT IFNULL(COUNT(*),0) FROM
        ( -- simulate except all operator with rowid
        SELECT ROW_NUMBER() OVER (PARTITION BY FLT.oid) AS RowNum, FLT.oid FROM
FINGERPRINT_LOCALIZE_TEMP FLT WHERE FLT.uid="$uid"
        EXCEPT
        SELECT ROW_NUMBER() OVER (PARTITION BY FO.oid) AS RowNum, FO.oid FROM
FINGERPRINT_OBJECT FO WHERE FO.flid=F.flid
        )
    ) AS dissimilarity, (SELECT IFNULL(AVG(weight),1) FROM
        ( -- simulate except all operator with rowid
        SELECT ROW_NUMBER() OVER (PARTITION BY FLT.oid) AS RowNum, FLT.oid,
OF.weight as weight FROM FINGERPRINT_LOCALIZE_TEMP FLT, OBJECT_FREQUENCY OF WHERE
FLT.oid =  OF.oid and FLT.uid="$uid"
        EXCEPT
        SELECT ROW_NUMBER() OVER (PARTITION BY FO.oid) AS RowNum, FO.oid, OF.weight
as weight FROM FINGERPRINT_OBJECT FO,  OBJECT_FREQUENCY OF WHERE   FO.oid =  OF.oid
and FO.flid=F.flid
        )
    ) AS weight
    FROM FINGERPRINT F
    WHERE F.modelid="$modelid" and F.buid="$buid" and
                (x between "$prevX" - "$smas_db_location_bound_meters" and "$prevX"
+ "$smas_db_location_bound_meters") and
                (y between "$prevY" - "$smas_db_location_bound_meters" and "$prevY"
+ "$smas_db_location_bound_meters")
                and (deck between "$prevDeck" - 1 and "$prevDeck" + 1)
    GROUP BY ROUND(F.X,"$smas_db_location_bound_rounding"),
ROUND(F.Y,"$smas_db_location_bound_rounding"), F.deck -- keep only x,y,deck rounded
by 10m bounding box
    HAVING dissimilarity < (SELECT COUNT(*) from FINGERPRINT_LOCALIZE_TEMP FLT
WHERE  FLT.uid="$uid")
    ORDER BY dissimilarity, weight, ABS(x - "$prevX") + ABS(y - "$prevY") +
ABS(deck - "$prevDeck")  ASC
    LIMIT 1; -- sort by newest match to oldest MATCH
```

7 SMAS Remote Evaluation and Field Study

Main author of the chapter: Paschalis Mpeis, UCY

7.1 Experimental Methodology

To assess the correctness and performance of the SMAS architecture and implementation, we carry out two experimental studies, in collaboration with T05.12 (Stena Flavia ro-pax), namely: (i) a **remote study in Section 7.2,** where we collect CV logs for the vessel based on video footage that the operator provided us. We use these logs to carry out a variety of tests and experiments in the laboratory (i.e., *in vitro*); (ii) an **on-board study in Section 7.3**, where we repeat certain evaluations scenarios on a real drill and compare the quality of results (i.e., *in vivo*). The results of our remote and on-board study show that SMAS is an extremely promising technology that promotes location-awareness and consequently situational awareness beyond the current state.

7.2 Remote Study on the Stena Flavia

7.2.1 Remote Study Methodology

In this section we provide additional details for our remote evaluation methodology.

Datasets: For the remote study we use 14 videos captured by Stena as part of the LASH FIRE project with a total size of 11GB (i.e., approximately 800MB per file) that had a total length of 1 hour and thirty minutes (i.e., each video was 6.55 minutes with standard deviation 2.83 minutes.) The videos capture a variety of areas like, Open ro-ro, Closed ro-ro, Weather deck, PAX public areas, Cabin areas but not any Crew areas. The focus of the deliverable has been mainly the Closed ro-ro, Open ro-ro and Weather deck, which are the decks 3 and 4 on the Stena Flavia, as such the bulk of footage comes from these areas. Additionally in the above areas there is no privacy concern for the usage of the camera localization system. Table 20 shows the overall status of the input dataset that will be used in the remote study.

Table 20: Video Files Used for the Remote Study

Video ID	Size (MB)	Length (min)	Deck	Area	Annotation	Filename
Video ID	**Size (MB)**	**Length (min)**	**Deck**	**Area**	**Annotation**	vessel02_10k (2.Loading operations Ventspils dk 3).mp4
V01	637.88	8.51	3,4	Closed ro-ro	ro-ro, elevator, stairs	vessel03_10k (3.Loading operations Ventspils dk 4 - 1st part).mp4
V02	640.88	8.51	4	Closed ro-ro, Open ro-ro	Empty, Loading	vessel04_10k (4.Loading operations Ventspils dk 4 - 2nd part).MP4
V03	1698.15	3.56	4	Closed ro-ro	Loading	vessel05_10k (5.Loading operations Ventspils dk 4 - 3rd part).mp4

V04	638.80	8.51	4	Closed ro-ro	Loading	vessel06_10k (6.Loading operations Ventspils dk 4 - 4th part).mp4
V05	643.77	8.51	4	Closed ro-ro	Loading	vessel07_10k (7.Loading operations Ventspils dk 4 - 5th part).mp4
V06	644.05	8.51	4	Closed ro-ro	Loading	vessel08_10k (8.Loading operations Ventspils dk 4 - 6th part).mp4
V07	385.17	5.12	4	Closed ro-ro	Empty	vessel14_10k (1.Evacuation routes from Assy A -part 2).mp4
V08	50.79	0.42	5	PAX public	Open Deck, Reception, ASSY A	vessel15_10k (2.Evacuation routes from Assy A).mp4
V09	546.31	7.3	5	PAX public	ASSY A, Restaurant, Open Deck, Reception, Passenger Deck	vessel16_10k (3.Evacuation routes from Assy B).mp4
V10	129.56	1.44	5	PAX public	ASSY B, Open Deck	vessel17_10k (4.Evacuation routes from cabins to Assy A).mp4
V11	476.28	6.29	6,5	PAX public, Cabin	cabin, stairs, restaurant, ASSY A loop	vessel18_10k (5.Localization Public spaces).mp4
V12	649.69	8.52	5,6	PAX public	restaurant, reception, open-deck, stairs	vessel19_10k (6.Localization walk dk 3).MP4
V13	3521.50	8.09	3	Closed ro-ro	cars, trailers, containers	vessel20_10k (7.Localization walk dk 4).mp4

Hardware: We use a 4K HDR 1ms Gaming Monitor with Eye-care B.I. Plus Sensor (i.e., Benq El2870u) that plays the video files on a Macbook Pro using the VLC tool. For the logging and localization tasks we use the Caterpillar S62 smartphone, which was selected during D06.4 - Background and testing of smart alert system of nearby responders as the experimental device apparatus. Particularly, the S62 features the following specs: Android 10, Qualcomm Snapdragon 660 (Qualcomm Kryo 260 CPU, Octa-core CPU, 64-bit, 1.95 GHz to 2.2 GHz, 4GB RAM, 128GB ROM, 4000 mAh non-removable Lithium-Ion battery, Image Signal Processor Qualcomm Spectra 160 image signal processor, 14-bit, 2x Image Signal Processor (ISP), Single Camera, MFNR, ZSL, 30fps: Up to 25 MP, Hybrid Autofocus, Optical Zoom, Qualcomm Clear Sight camera features, Zero Shutter Lag. Even though the description of the capturing device sounds very specific, the system works well with any smartphone having a capable camera.

Tuning Remote Logging: The video playing tool was configured in the Logging experiments at "Medium" speed to provide our camera system enough time to capture the various objects. Generally, due to varying lighting conditions as well as inherent pixelation and discretization of a computer monitor output we observe that our logger and localization engines did not recognize as many objects as they would recognize if the input came directly from the camera. As such, the logging and localization output is much worse that what it would be in a real scenario. On the other hand, logging in front of the PC gives us more time for a thorough logging process (i.e., replaying a video and relogging if necessary). Nevertheless, it provides us with a realistic environment to assess the performance of our developed algorithm.

7.2.2 Logging Time Evaluation

Purpose: The purpose of this evaluation is to observe how much effort is necessary to collect object sets through the SMAS Logger to enable localization on a vessel. Our effort metric is expressed mainly in time, but we do also measure the number of objects that were collected per deck.

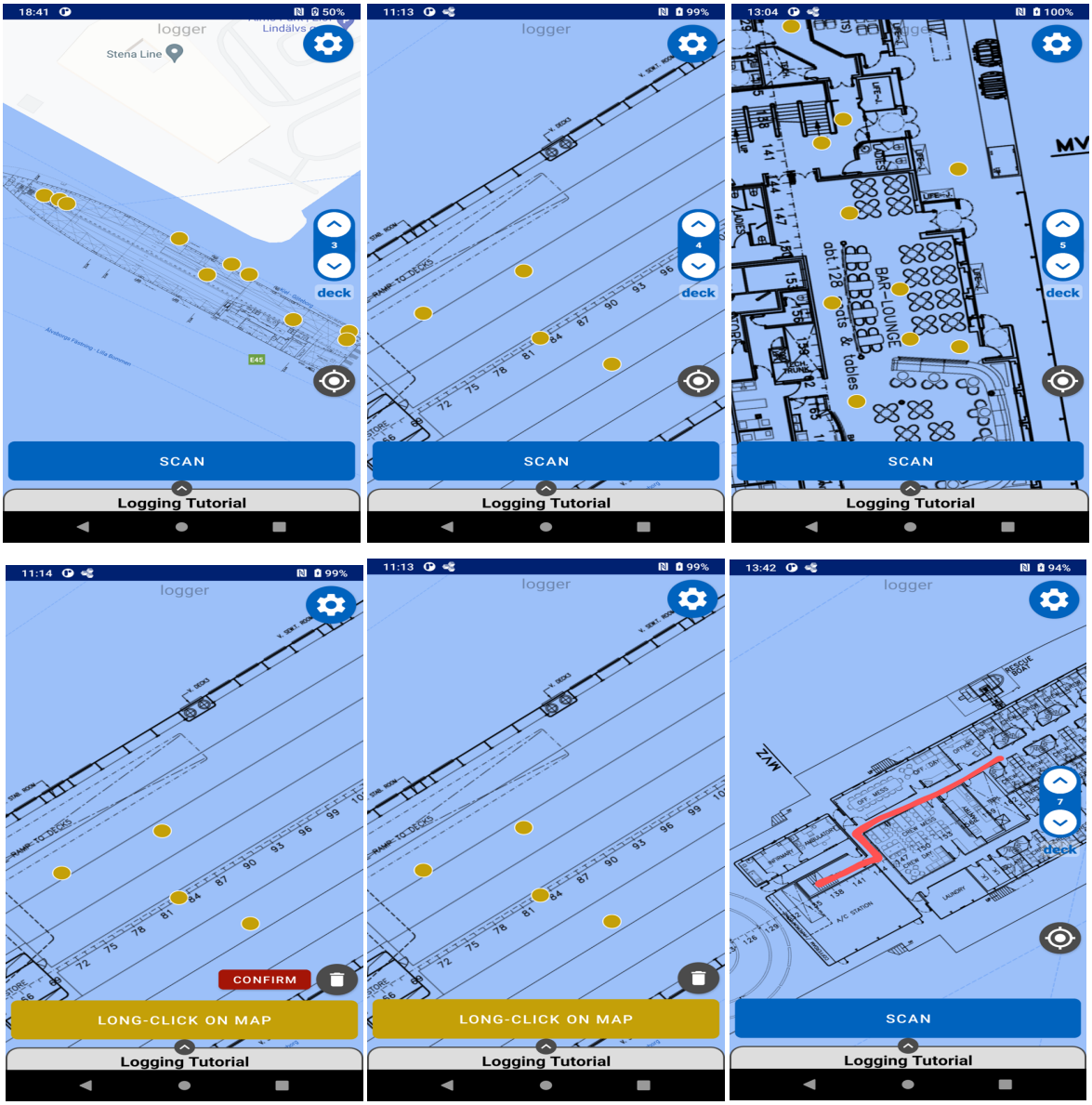


Figure 29: *Collecting and Managing Fingerprints with the SMAS Logger. All results are stored on the SMAS service and used for subsequent localization requests.*

Description: We analyzed the 14 videos of the Stena Flavia ro-ro passenger ship and for each video we have made two passes. With the first pass, we acquaint ourselves with the indoor and outdoor spaces of the vessel, by carefully observing and mapping key points in the video with points in the deck plan. ***Figure 30*** shows an example of how the mapping from real-world footage to an actual trajectory on the path is performed. From the surrounding objects (chairs and round tables) we understand that the user of the recording device was initially in the restaurant (first picture). Then, by closely following the

user direction and movement we can reconstruct the path that the user was following. In this example, the user went outdoors on the right-hand side of Deck 5 and made a U-turn in front of a lifeboat. Particularly useful were signs on various areas that helped us find where each area of the video is located on the map (for the purpose of correctly registering the CV logs).

Figure 30: *The modelling of a vessel (for the indoor and outdoor spaces) does not require physical presence. To accomplish this remotely, we perform an initial pass over the footage so we can understand how the real word spaces (shown in the first 3 pictures) map to the vessel's deck plan (shown in the last picture).*

Once we understood the indoor and outdoor spaces of the vessel, we perform the second pass where we do the actual mapping. During this pass, we use the SMAS Logger in a "Demonstration Mode", which simply identifies objects continuously. This allows us to detect which spots within a space contain *strong fingerprints*. These are a collection of recognized objects that have a high probability of being unique in the current deck or even in the entire vessel. ***Figure 31*** shows a few examples of such fingerprints. Once we empirically identify such *strong fingerprints*, we switch to the logging mode which in turn assigns to these objects a physical location on the map.

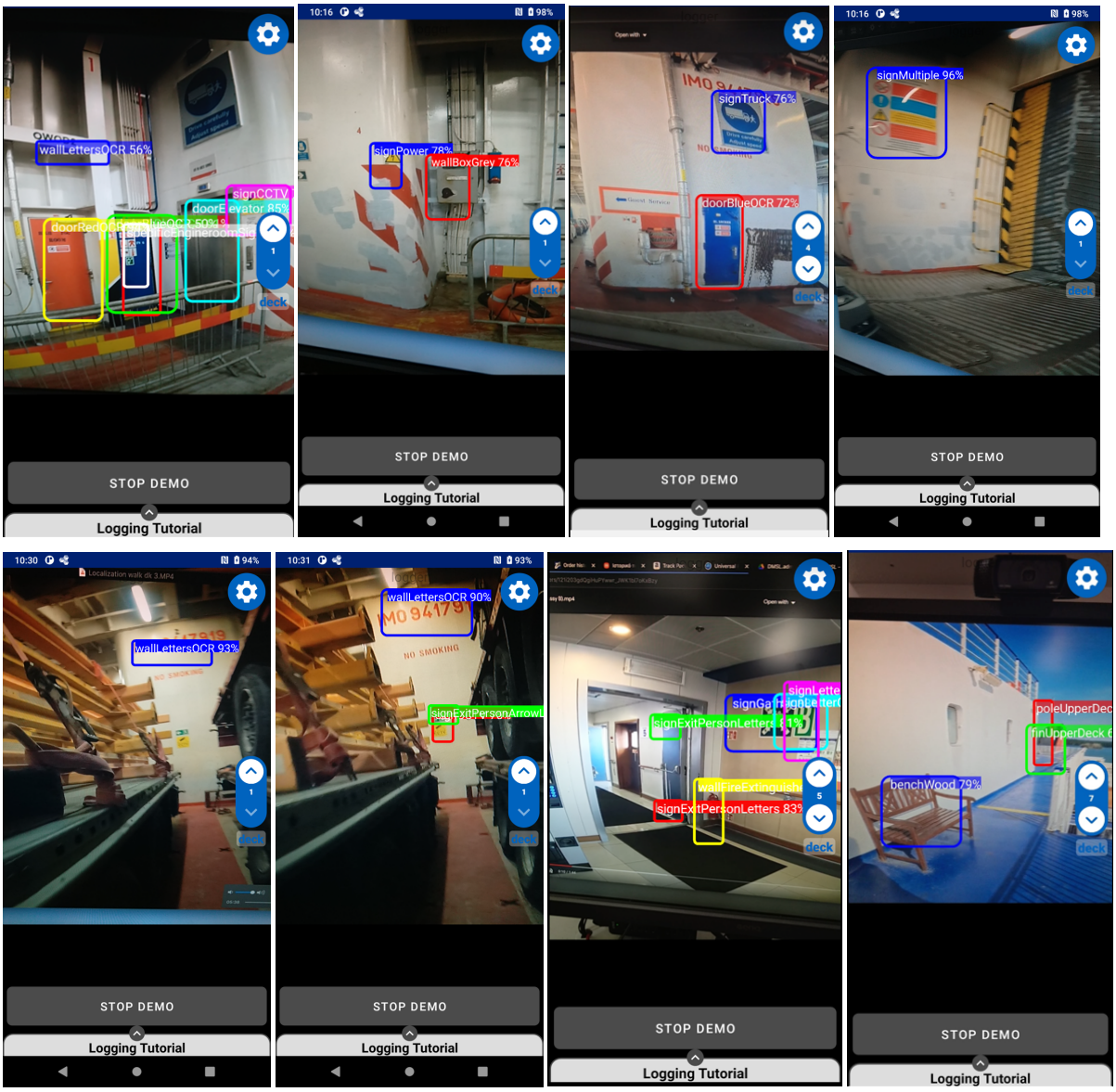


Figure 31: *Examples of strong fingerprints on different scenes in the ro-ro and PAX public space. Each of those combinations of objects are unique for the entire vessel. Therefore, the SMAS Logger is mapping those to physical coordinates (i.e., modeling process), so they can be used for uniquely identifying a user's location.*

We were even able to use on-vessel footage that was take for unrelated purposes, like 4 videos that show the evacuation routes, or 7 videos that show the loading operations of trucks. Some of the videos contained fingerprints that were already modeled, so we did not have to create new entries for those. We could easily check for such cases by testing localization before storing a new fingerprint.

Results: In ***Figure 32***, on the left side, we show the logging time required to model the Stena Flavia vessel from a remote location. Deck 5 required the most time by far, with a logging time of a bit more than 2 hours. This was the most diverse deck of the vessel in terms of room types, consisting of 4 restaurant areas, a bar, a lounge space, a reception hall, 2 outdoor spaces, a car storage space,

amongst others. Deck 4 required about 10 mins, while Deck 3 and Deck 7 required less than 30 minutes. Deck 6 did not have many fingerprints and required just 10 minutes. For each spot it required roughly 3 mins.

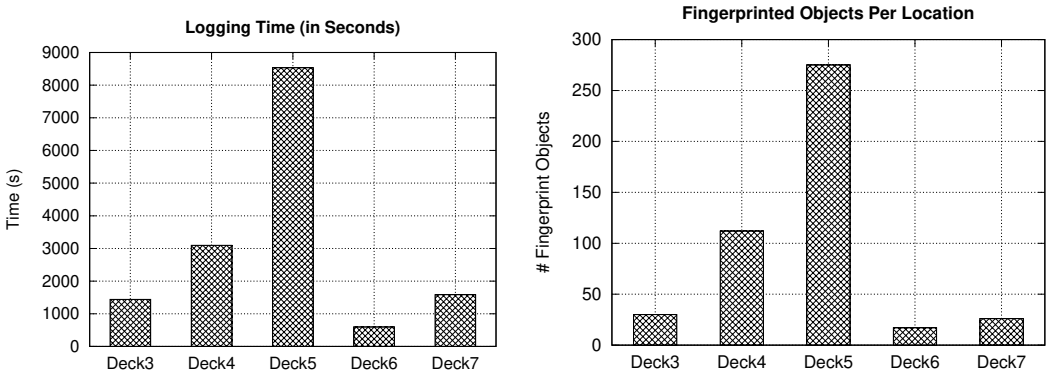


Figure 32: *The left plot shows the time that was required to perform a remote logging per deck on Stena Flavia using footage from 14 videos, while the right plot shows the number of objects that were fingerprinted per deck.*

On the right side of in ***Figure 32***, we visualize the fingerprint objects that were captured during the modeling of Stena Flavia. In total 460 objects were stored in 5 different decks. Those were clustered in 81 geographic locations. More than half of the objects were assigned in Deck 5, which was the most diverse deck in terms of space types. Deck 4 had a bit more than 100 objects, while the remaining 3 decks had 25 fingerprinted objects on average.

On ***Figure 33*** we show the 5 floors that have collected fingerprints through the remote study. The depicted heatmaps are useful to the installation team to understand where additional collection of objects is necessary. This can be in the future associated with the accuracy estimation framework ACCES we discussed in D06.4 - Background and testing of smart alert system of nearby responders as part of our prior work.

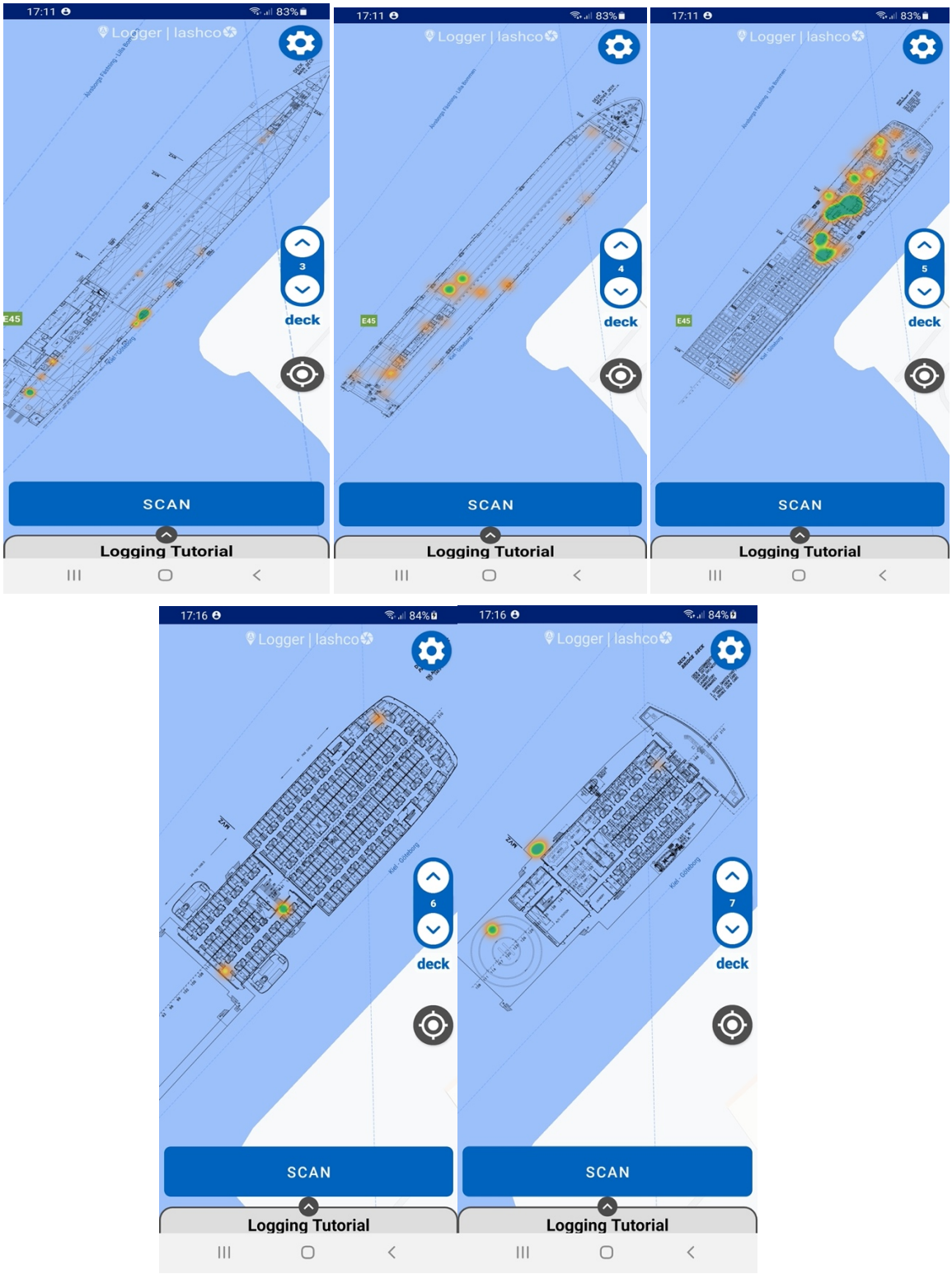


Figure 33: *Seeing the fingerprints mapped on the SMAS Logger interface helps the installation team understand where additional object collection is necessary. For example, the above shows that deck 5 clearly has sufficient logged objects to enable accurate localization.*

We next present the OBJECT_FREQUENCY table using the WGS84 11-meter clustering method we proposed for the logged data. This table shows us the rarest objects (from rare to frequent ranking) on the Stena Flavia based on our Logging effort.

Table 21: OBJECT_FREQUENCY View of the Remote Study Logging Task. The figure shows objects ranked by how rare they were with the WGS84 11 meter clustering method. The below weights are taken into account in the localization ranking function .

Object ID (oid)	Object Description	Weight (rare to frequent ranking)
98	signBathroomDisabled	0.00526315789473684
95	signFireSafetyLever	0.00526315789473684
93	wallSingleDigitOCR	0.00526315789473684
92	specificTankDeck	0.00526315789473684
91	signNumberOCR	0.00526315789473684
90	signArrow	0.00526315789473684
82	floorPatternCorridor	0.00526315789473684
76	specificPlaygroundHouse	0.00526315789473684
74	specificRestaurantFreezer	0.00526315789473684
72	stairsInterior	0.00526315789473684
68	floorPatternReception	0.00526315789473684
67	ceilingParking	0.00526315789473684
64	signMultiple	0.00526315789473684
60	specificEngineroomSigns	0.00526315789473684
57	doorElevator	0.00526315789473684
47	doorRedOCR	0.00526315789473684
35	redLineXdouble	0.00526315789473684
33	lightUpperDeck	0.00526315789473684
24	handleWood	0.00526315789473684
23	restaurantChair	0.00526315789473684
22	restaurantTableRound	0.00526315789473684
83	signExitLettersOnly	0.0105263157894737
79	wallCorridorCabinLight	0.0105263157894737
78	handleIron	0.0105263157894737
77	restaurantTheatreChairs	0.0105263157894737
62	signPower	0.0105263157894737
61	wallPlugBlue	0.0105263157894737
58	ramp	0.0105263157894737
45	doorBlueWindow	0.0105263157894737
32	finUpperDeck	0.0105263157894737
31	poleUpperDeck	0.0105263157894737
28	benchWood	0.0105263157894737
27	slotMachine	0.0105263157894737
26	deskInfo	0.0105263157894737

21	restaurantDivider	0.0105263157894737
15	trashcanYellow	0.0105263157894737
6	wallNumberOCR	0.0105263157894737
97	signFireExtinguisher	0.0157894736842105
53	wallFireExtinguisherEmbeddedOCR	0.0157894736842105
13	trashcanMulticolour	0.0157894736842105
12	trashcanRed	0.0157894736842105
5	wallLettersOCR	0.0157894736842105
2	signTruck	0.0157894736842105
99	symbolNoSmoking	0.0210526315789474
96	signFireGlassBreak	0.0210526315789474
94	signLifeSaver	0.0210526315789474
88	signExitPersonArrowLeft	0.0210526315789474
71	handleDoorBlueInterior	0.0210526315789474
50	wallPatternYellowBlack	0.0210526315789474
19	wallEmergencyButtonRed	0.0210526315789474
10	stairsBlueIron	0.0210526315789474
73	restaurantChairLegs	0.0263157894736842
70	trashcanInterior	0.0263157894736842
55	fireExtinguisherBoxed	0.0263157894736842
46	doorWood	0.0263157894736842
44	doorBlueOCR	0.0263157894736842
30	lifesaverRing	0.0263157894736842
18	wallFireExtinguisher	0.0263157894736842
3	signGather	0.0263157894736842
75	deckOutsideFence	0.0315789473684211
59	signCCTV	0.0315789473684211
49	wallPatternRedWhite	0.0315789473684211
4	signExitPersonLetters	0.0315789473684211
69	signLetterOCR	0.0368421052631579
43	doorBlue	0.0368421052631579
52	wallFireExtinguisherEmbedded	0.0473684210526316

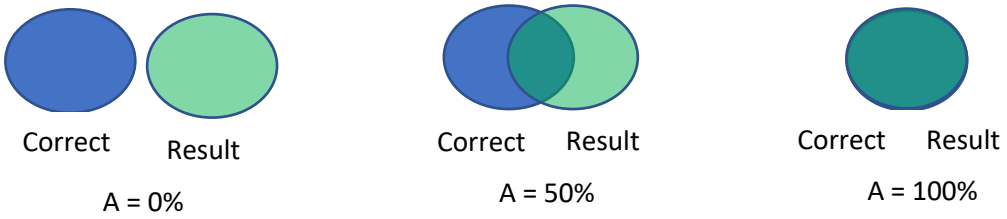
7.2.3 Scene Localization Accuracy Evaluation: The SG Algorithm

Purpose: The purpose of this evaluation is to assess the localization accuracy of the Surface Global (SG) Algorithm we designed and implemented. Particularly, our aim is to observe whether the SMAS CV Localization app can distinguish with high accuracy among 15 scenes we selected out of our video input set in a remote study context. In the field study of Section 7.3 we aim to verify that the results in an in-person study will be even better, mainly because the camera system will be able to recognize more objects vs. from a screen, which suffers from discretization and lighting issues.

Metrics: We use the **Localization Accuracy (A)** as the metric for our evaluation. Generally, accuracy measures the closeness of measurements to the true value of the quantity being measured. In our case, we measure in a binary manner whether the returned location is correct to the true value by the means of an expert user how judges based on the map and the scene the validity of the result. Specifically, the expert user carrying out the result judges with true or false whether the correct location has been returned for a given scene. The above is repeated a number of times (in our experiments repetitions is set to 10). Below we provide the definition of our accuracy using a set theoretic notation, where Result indicates the returned location and correct location.

$$Localization\ Accuracy$$

$$A = (\sum_{i=1}^{n} \frac{|Result_i \cap Correct_i|}{|Correct_i|})/n * 100$$



Description: For this experiment we have used footage from 6 different videos. We have used 15 different scenes in total, with two thirds concerning cargo/ro-ro areas and one third concerning passenger areas (PAX). We made this decision as T06.10 and D06.6 was mainly focused on the former areas. *Table 22* lists the 7 different parameters that each scene had, as well a short description. These include whether the scene was used in training or not, the deck number, the video source, the lighting conditions of the scene, the objects that were contained by the scene (e.g., cars, tracks, passengers, or mixed), the density of those objects in the scene, and finally whether the camera that captured the footage was moving or not.

Scenes:

In this paragraph we describe the 15 scenes (of ***Table 22***) in more detail.

- In **S01** the user was in front of the lift in Deck 3, where many objects have been recognized.
- In **S02**, the user was standing in front of the Cargo Office in Deck 3, and a few objects have been recognized.
- In **S03**, the user was in front of walls that are surrounding the funnel, recognizing a few labels on that wall.
- In **S04**, the user was walking between tracks, recognizing some labels on the front of the vessel.
- In **S05**, the user was recognizing some outdoor equipment.

- In **S06** and in **S08**, the user was crossing between the lanes that are used for parking tracks or cars, recognizing some labels and doors on the outer wall of the vessel.
- In **S07**, the user was walking in the front of the vessel recognizing some labels and doors on the wall.
- In **S09,** the user was walking between tracks and the left-outer wall of the vessel, recognizing some equipment.
- In **S10,** the user was freely walking in Deck 4, recognizing some patterns, bins, labels, and other objects in the wall.
- In **S11,** the user was walking in front of the Information Desk in the vessel's Reception, recognizing it.
- In **S12,** the user was walking in the restaurant/lounge bar area, recognizing specific table and chair types.
- In **S13**, the user was outdoors, in the Open-Deck area, recognizing life-saving equipment.
- In **S14,** the user was passing in front of in the main stairs close to the restaurant in Deck 5, recognizing them.
- Finally, in **S15,** the user was in the cabin rooms of Deck 6, recognizing some labels and some lights that are present only in cabin corridors. Video V13, contained around half of those scenes, while the remaining scenes were distributed to 5 other videos.

Table 22: Summarizing the 15 scenes that were used in the localization experiments using 6 different parameters. The first column (#) shows the Scene identifier (i.e., Sxx). The next two columns show the video footage that was used, as well a Short Description. The remaining 6 columns present parameters that differentiate the scenes between them.

#	Video	Short Description	Deck	ML Trained	Type	Lighting	Density	Mobility
S01	V13	Lift	3	NO	mixed	lights	full	static
S02	V13	Cargo Office	3	NO	mixed	lights	semi-full	static
S03	V13	Funnel (front)	3	NO	mixed	lights	semi-full	moving
S04	V13	Between-Trucks-180	3	NO	mixed	low	full	moving
S05	V14	outdoor equipment	4	YES	mixed	high	full	static
S06	V14	cross walk	4	YES	mixed	lights	full	moving
S07	V14	wall walk front	4	YES	mixed	low	full	moving
S08	V02	cross walk	4	YES	mixed	lights	full	moving
S09	V14	wall left	4	NO	truck	low	full	moving
S10	V14	free walk (middle)	4	yes	empty	day-light	empty	moving
S11	V14	Reception	4	yes	passenger	high	semi-full	static
S12	V09	Restaurant	5	yes	passenger	lights	empty	moving

S13	V09	Open Deck	5	yes	passenger	high	empty	static
S14	V12	Stairs	5	yes	passenger	lights	empty	moving
S15	V11	Cabins	6	yes	passenger	low	semi-full	moving

Results & Discussion:

Figure 34 shows the success percentage of 150 localization attempts over the 15 above scenes, as well the success average. For each scene we performed 10 localization attempts, and we calculate Localization Success percentage. A localization attempt is considered as successful when the user was able to localize within 10 meters of the ground truth. For about two thirds of the scenes, we were able to achieve a remarkable 80% success, while for 3 scenes we achieved a perfect score. **On average, we achieved an 80% localization success!**

Scenes S04, S09, and S15 were the ones with the lowest percentage, that was still 60%. In S04, the user was walking in a lane that had high-height tracks on both sides and was not able to recognize a necessary number of objects before attempting to localize. There is a similar scenario in S09, with the only difference that the user was walking in a lane that had tracks on the one side and the vessel's wall on the other. Lastly, in S15, the lighting and stabilization conditions for the footage were not ideal, hence the CV system was not able to readily identify the objects. The most contributing factors in these 3 cases were the lighting conditions, monitor reflection (due to remote execution of this experiment), as well the naturally low number of objects in those areas. The on-board study verifiesx this claim.

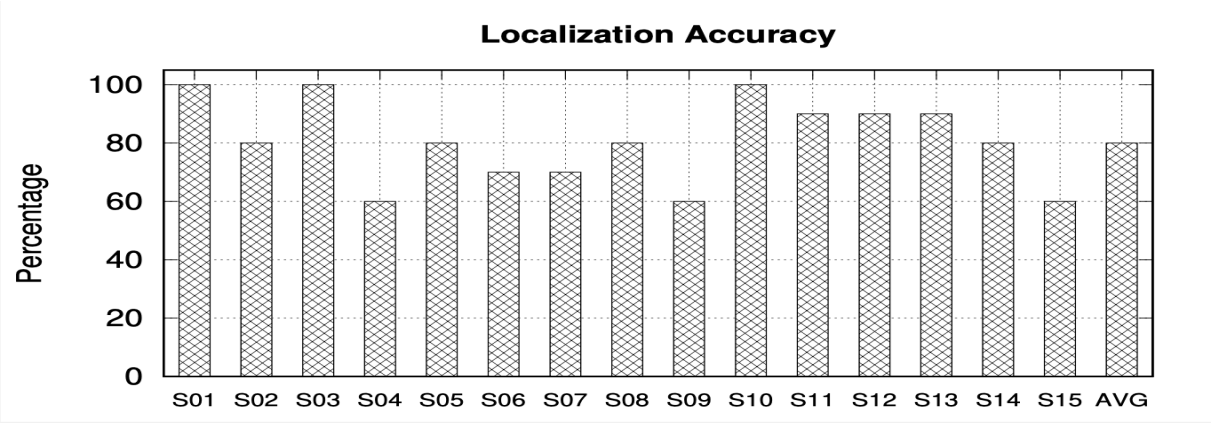


Figure 34: *This plot shows the localization success percentage for each of the 15 Scenes (labeled S01-S15), as well the average. In more than two thirds of the scenes we were able to achieve between 80%-100% accuracy, with an average of 80%, in a total of 150 localization attempts.*

7.2.4 Trajectory Localization Accuracy Evaluation: The SL Algorithm

Purpose: The purpose of this evaluation is to remotely assess the Computer Vision localization system when localizing on a particular trajectory. We have used two different trajectories.

Description: For this experiment we have used footage from 2 different videos, V09 and V13 and followed two different trajectories. Each trajectory had 5 points, and for each point we have performed localization for 5 times.

Figure 35: *Showing the Trajectory 1, that was in Cargo Areas in Deck 3. The first image shows the 5 points that were followed, along with their order in the trajectory. The remaining images show the objects recognized at each point of the trajectory.*

In the first trajectory, shown in ***Figure 35*** the user was in Deck 3, right outside the engine room (T01). Then the user has moved towards the ramp of the vessel (T02), and then at the Cargo Office (T03). Then, the user has moved towards the center of the vessel (T04), and finally has crossed to the right side of the vessel, close to the Pilot Door (T05).

Figure 36: *Showing the Trajectory 2, that was in Passenger Areas in Deck 5. The first image shows the 5 points that were followed, along with their order in the trajectory. The remaining images show the objects recognized at each point of the trajectory.*

In the second trajectory, shown in ***Figure 36*** the user was in Deck 5, and just enter the Drivers Restaurant (T01). Then the user has moved in front of the playground (T02), before going in the main restaurant (T03). Then the user has moved to the Lounge Bar (T04), before finally going to the Assembly Point A (T05).

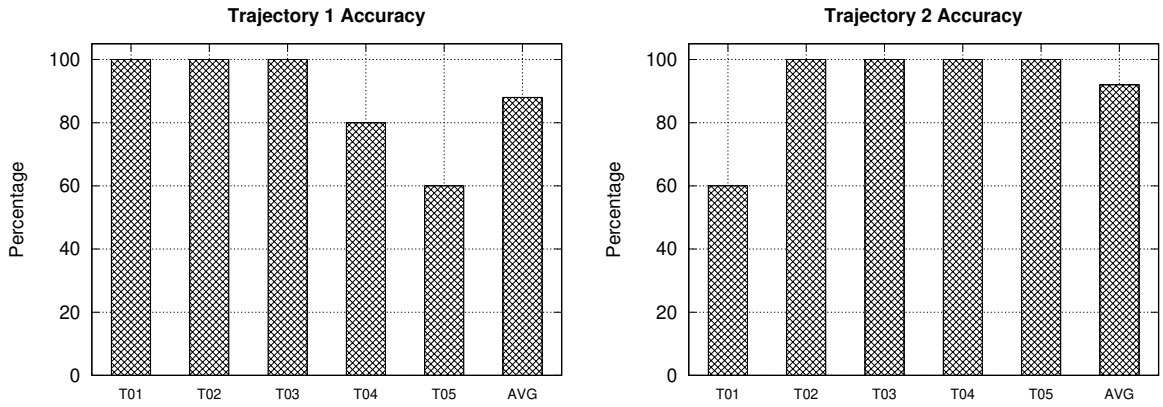


Figure 37: *Showing the Trajectory Accuracy results when following Trajectory 1 (left side) and Trajectory 2 (right side). Each trajectory had 5 points and we have repeated the experiment 5 times. The average accuracy in both trajectories was an impressive 90%.*

Figure 37 shows the Trajectory Accuracy results for the 2 trajectories. The average accuracy percentage was 88% and 92% for Trajectory 1 and 2, respectively. We have achieved a perfect score for 3 out of the 5 points of Trajectory 1, and 4 out of the 5 points for Trajectory 2. The lowest point in Trajectory 1 was T05, at 60%, where the user was not able to recognize some floor objects due to fast video movement. The lowest point in Trajectory 2 was T01, and similarly at 60%, where the user was capturing only one out of the two objects. As a result, the localization algorithm pointed the user to a nearby location, instead of the Drivers Restaurant.

7.2.5 Scalability Evaluation

The purpose of the third evaluation was to assess the scalability of the localization algorithms. In particular, our aim was to observe the localization response time as the number of fingerprints increases. The remote study had around 0.5K object fingerprints mapped to 81 individual locations. In this experiment we uniformly and randomly generate 7 different object fingerprint datasets at much larger scales and observe the response time of the localization algorithm. We chose the SL algorithm with a random prior point and randomly chosen objects, as this exposes the more complex case for our algorithm (i.e., in case SL does not yield any result, the SG algorithm is executed.)

In Figure 38 we visualize the query response time in milliseconds (ms) for the 7 datasets. As shown, Surface scales linearly with the dataset size. For the real dataset (1k points), we required less than 10ms. For 10k points we required about 50ms, which is sub-linear. Between 10k and 50k the increase is linear, as we observe ~250ms. This is within the latency of the CV object detection system, so we've verified that the localization algorithm will not be on the critical path of the evaluation even for very large Fingerprint Database scenarios capturing complete vessel spaces. For reference we mention that the CV object detection system requires on the S61 around 400-500ms and 200-300ms on the S62. For more powerful smartphones this can be less than 100ms [9], but localization is not anticipated to run so frequently.

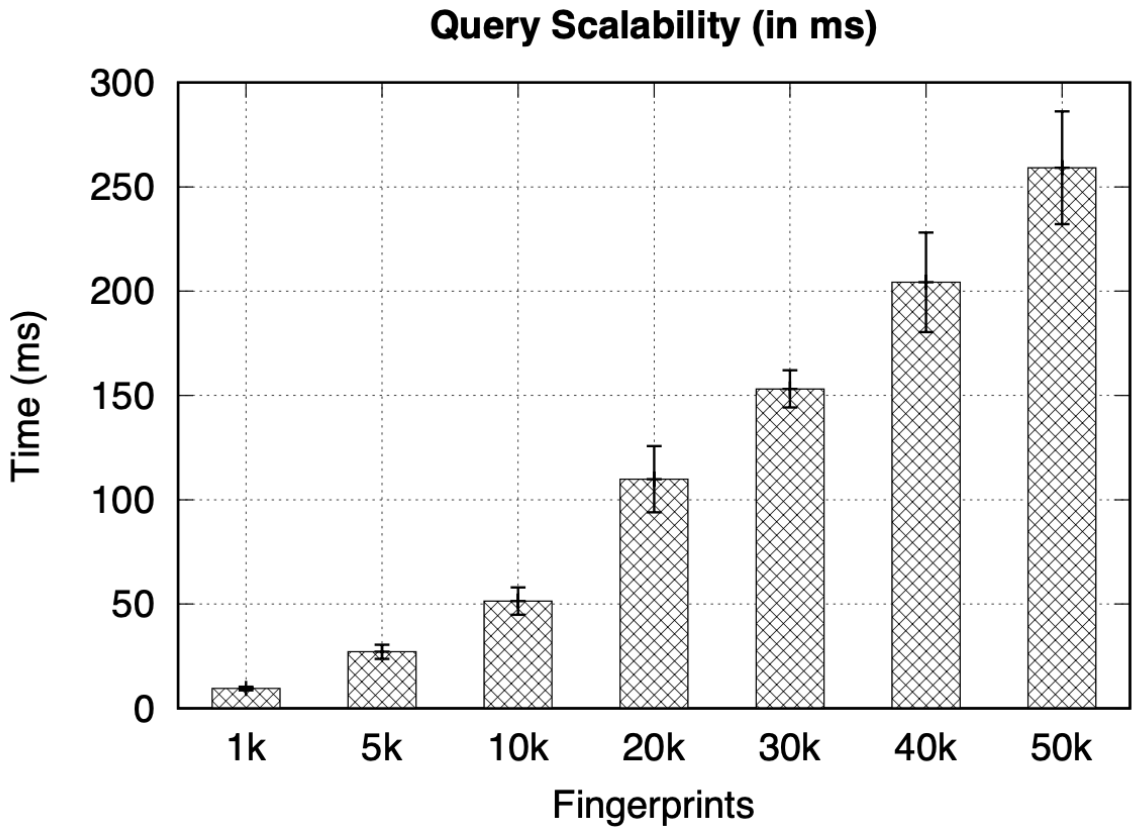


Figure 38: The Surface localization algorithm scales linearly with the increase in the size of the Fingerprint Database (FDB).

7.2.6 Chat Nearest Neighbor Algorithms Evaluation

In this section we carry out a performance study of the different messaging methods of the SMAS system. The goal of this evaluation is to highlight the importance of granular, location-aware filtering system when it comes to tackling emergencies.

Figure 39 compares the latency in ms for delivering a message using different methods that restrict the outreach of each message. The first method, shown as BBox, uses a bounding box, surrounding a user, to limit its outreach. The second, shown as KNN, reaches the k nearest neighbors of a user. The following two, shown as Deck3 and Deck1, message only the users that are on Deck 3 and 1, respectively. Finally, the method named All, messages all the users, regardless of their location.

During this experiment, we have deliberately kept the number of users small. This highlights that even with a few users the message distribution latency is significantly affected. We have observed an average latency of around 65ms when using the bounding box method. This method reaches any user that is within a configurable radius of the user that is broadcasting a message. When restricting the outreach to any user on Deck3, we have observed 12% less latency. This was due to the fact that fewer users were on Deck3 than there were within the bounding box. When using KNN, we observed 87.5ms average latency. This is just 4% less than messaging all the users in Deck1, which was the most crowded deck. Finally, when messaging all users, a latency of around 121ms was observed, which is 2x and 33% more when compared to the least and most crowded decks, respectively.

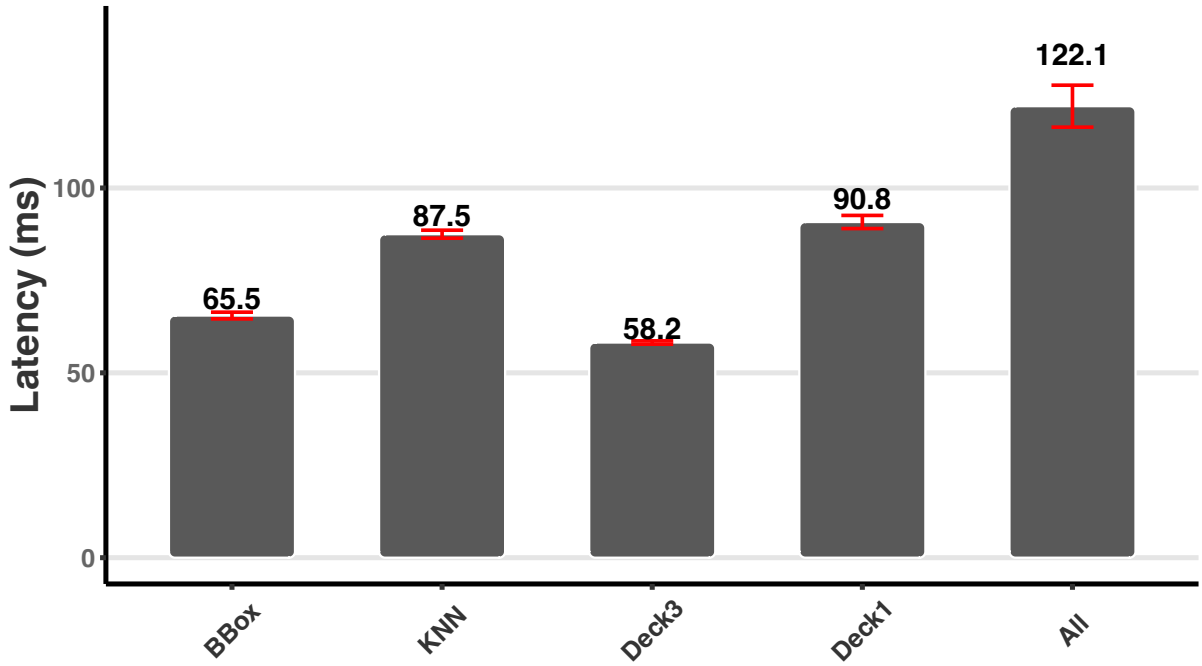


Figure 39: *Comparing the latency (in ms) when different messaging methods were used. For each method, we have sent 1000 messages. The error bars show 95% confidence intervals.*

These results highlight the importance of having an effective filtering system that is location-aware. When tackling emergencies, a combination of these methods is needed, in order to both limit the messages to the relevant people, but also to deliver them as soon as possible.

7.3 Field Study on the Stena Vessel

In this section we describe the field study that we performed on board the Stena Flavia vessel. The study was conducted to evaluate the Anyplace Logger and the SMAS system (mobile application as well the backend server) on real life scenarios. The vessel departed from Liepāja, Latvia, at 8PM and arrived in Travemünde, Germany around 6PM local time. We boarded on the vessel as soon as it has docked to the port, around 4PM. During the first day, on the 17th of August 2022, we have performed additional logging (see Section 7.3.1) to improve the indoor localization accuracy, and run experiments in low light conditions (see Section 7.3.3). During the second day, on the 18th of August, we have performed localization tests as well a drill test that involved several crew members, including the captains (see Sections 7.3.2 and 7.3.4, respectively). The on-board study has provided valuable feedback on the developed systems. The remaining of this section describes the field study and provides some concluding remarks and future ideas.

7.3.1 Logging Time Evaluation

During the first day of the field study, we have boarded in as soon as the vessel has arrived, around 4 PM. This was done to cover different scenarios in cargo and car spaces, such as localizing in unloaded decks, semi-loaded decks, and fully loaded decks. The additional logging (on top of the remote logging) was done in the following decks: 3, 4, 5, and 6.

Purpose: The purpose of this evaluation is to compare the logging process when performed on an actual vessel, versus when performed remotely with video footage.

Results: In ***Figure 40***, we show the time it took to perform additional logging (in addition to the remote study logging), while on-board the Stena Flavia vessel. Similarly with the remote study, deck 5 required most of the time, mainly for two reasons. First, it is the more diverse deck in terms of indoor spaces (see Section 7.2.2). Additionally, we were able to extend the logging to more areas that were not part of the video that was received for the remote study. It required a bit less than 2 hours, despite logging around 2.5 times more objects when compared to the remote study. Deck 3 and deck 4 required from 20 to 25 minutes approximately, while deck 6 required about 15 minutes. Less than 3 hours were required for the whole logging process, while the time needed per location was just 1 minute. This was 3x times less than doing it by video footage. The difference is due to the additional overhead that was needed in the remote study to acquaint ourselves with the indoor spaces and directions within the vessel.

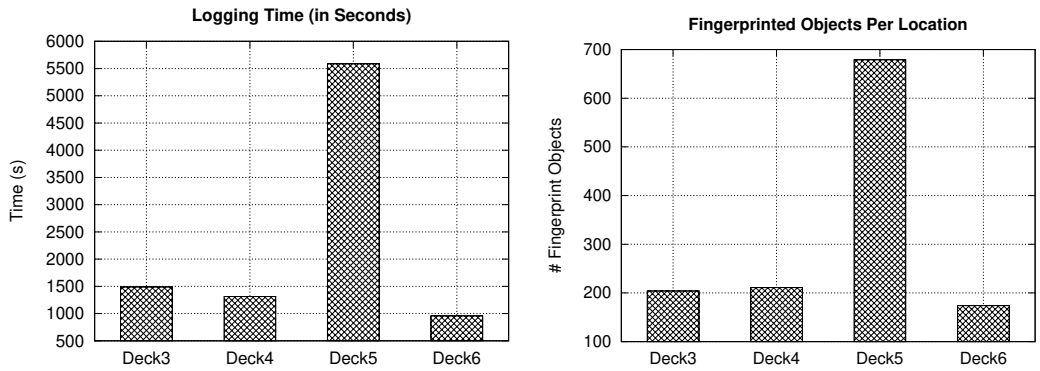


Figure 40: *The left plot shows the time that was required to perform logging per deck, while the right plot shows the number of objects that were fingerprinted per deck. Both experiments were performed during the field study, on board the Stena Flavia vessel.*

On the right side of in ***Figure 40***, we visualize the fingerprint objects that were captured during the logging that was performed on-board the Stena Flavia vessel. More than 1,200 objects were stored, about 2.7x more in comparison with the remote study. These objects were clustered in 176 different geographic locations on 4 different decks. This is because the logging operation was not limited to video footage, like in the remote study, but also because the physical presence allows better object recognition, due to the reasons explained in Section 7.2. Deck 5 had the most objects (more than 650 recognitions), as we were able to operate on a wider area, in contract with the limited footage of the remote study. The remaining 3 decks had around 300 objects.

7.3.2 Localization Accuracy Evaluation

One of the actions performed during the second day of the field study was the localization accuracy evaluation. In this experiment we have localized on-board the Stena Flavia vessel using the same scenes that were used in Section 7.3.2. ***Figure 41*** shows the offline localization procedure in action.

Purpose: The purpose of this evaluation is to assess the localization accuracy in real conditions on an actual vessel. We have repeated the experiment of Section 7.2.3, using the same 15 scenes while on board the Stena Flavia vessel.

Metrics: The same metrics with Section 7.2.3 have been used to provide a fair comparison with the remote study.

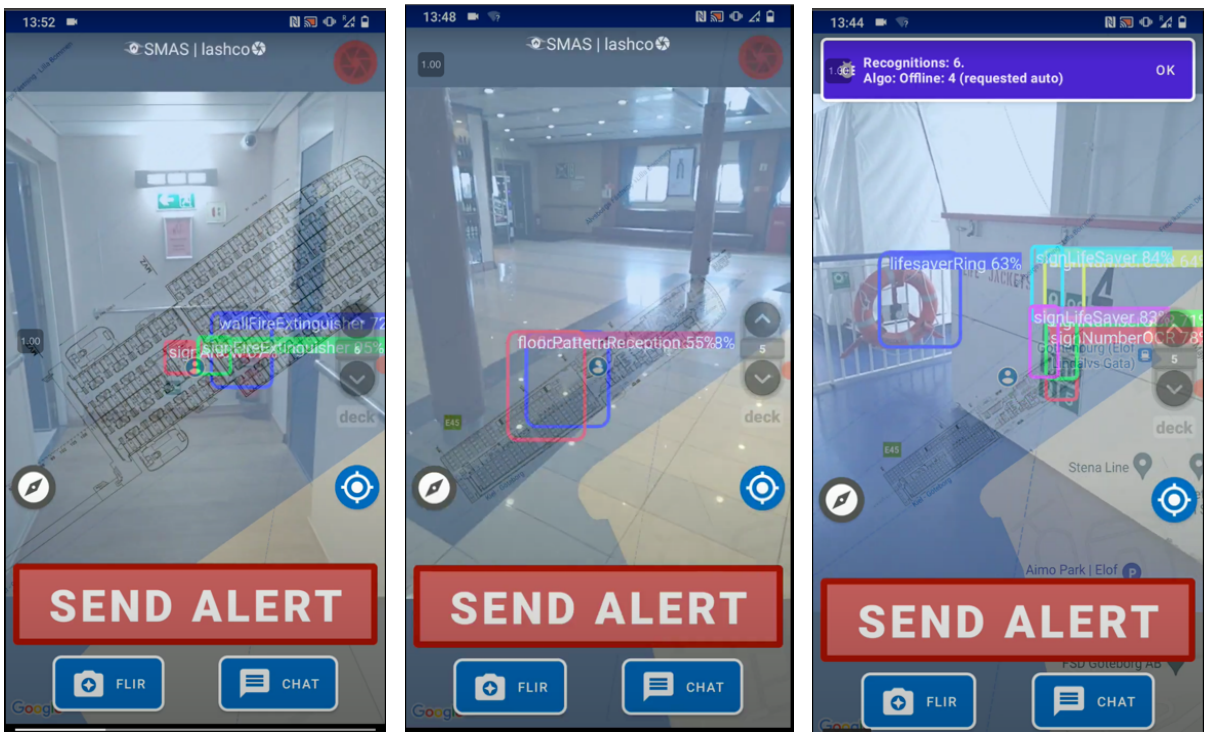


Figure 41: *Offline localization in action on-board the Stena Flavia. From left to right, we show the scenes 15, 11, and 13.*

Results & Discussion:

Figure 42 shows the success percentage of 150 localization attempts of the same 15 scenes that were described and used in Section 7.2.3. For about half of the scenes we were able to achieve more than 80% of successful localization attempts. This is a decline from the 2/3 of localization attempts of the remote study. Regardless, the perfect localization attempts (with an 100% score) were increased by 66%. This increase can be due to the absence of monitor reflection as well the higher image quality that is due to being physically present on the vessel when compared to the video footage.

Scene 6 had only a 10% localization success. This was due to the different placement of cargo vehicles on-board, when compared to the remote study. As a result, only a very steep viewing angle was possible, which influenced this scene's outcome as well the experiment's average. Scene 5 also had a dramatic decrease. This was due to increased and direct sunlight, but also due to equipment wear (i.e., some equipment colors were significantly washed out). This signals that additional training that includes different angles as well different lightning conditions, can be used as an improvement of the model (i.e., LASHCO). Scenes 7, 8, and 9 also had an accuracy decrease. However, those had lower accuracy in the remote study also, when compared with the other scenes.

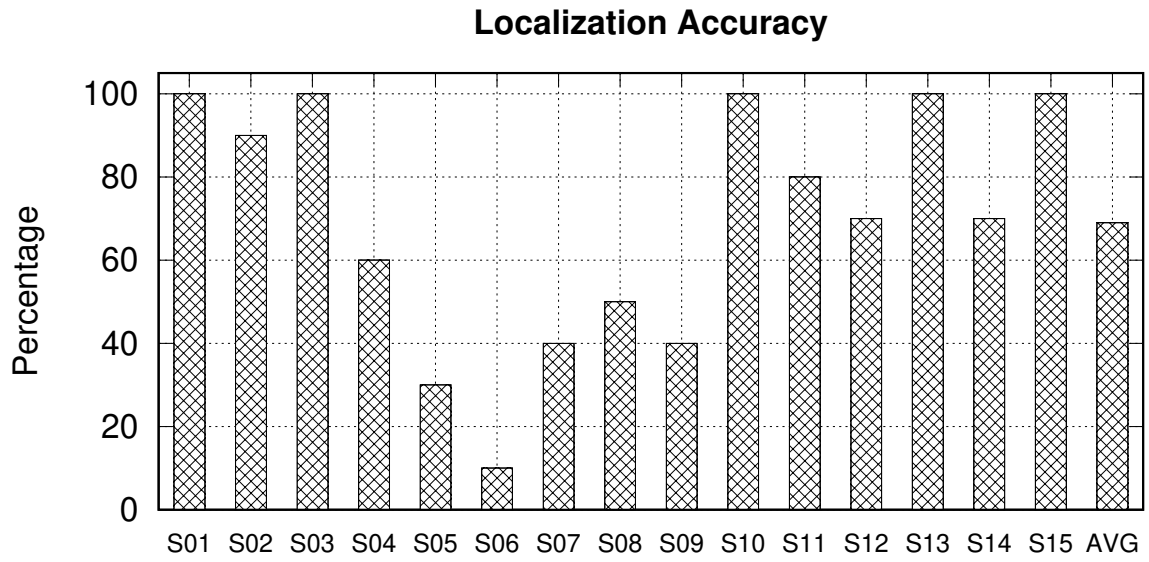


Figure 42: *This plot shows the localization success percentage of the 15 Scenes (labeled S01-S15; described in Section 7.2.3), as well the average. This experiment was performed on-board the Stena Flavia vessel. For around half of the scenes we were able to achieve between 80%-100% accuracy, with an average of around 70%.*

7.3.3 Battery and Low-Light Evaluation

In this section we describe the battery experiment, followed by a low-light evaluation experiment.

Purpose: The purpose of this evaluation is to assess on a real-life scenario the performance of the proposed approach in terms of power consumption as well low-light conditions.

Battery experiment: In this experiment we have recorded the battery consumption, in terms of battery percentage drop, for each deck. We have kept separate statistics for logging and localization. In essence, the same operations are performed under the hood for both operations (i.e., Deep Learning inference).

All but the logging of deck 5 have caused a 6% or less battery drop. Logging of deck 5 lasted for about 2 hours, which is significantly more than the logging or localization time that was required for each other deck. In total, logging caused a 36% battery percentage drop while localization caused a 14% battery drop. The difference between the two methods was only due to the extra time that is needed when the logging phase is performed. Logging is a one-off procedure, until the indoor spaces of a vessel or a building is adequately mapped.

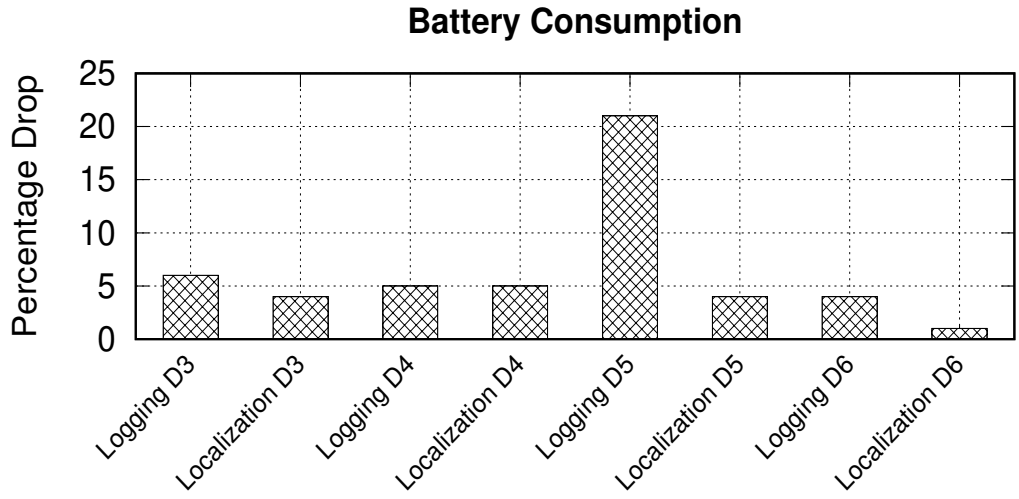


Figure 43: *Showing the battery consumption during the logging and localization experiments. The results are shown per deck for each method (i.e., logging and localization), and the bar represents battery percentage drop on the CAT S62 smartphone devices. In total the logging process caused a 36% drop while the localization process caused a 14% battery drop.*

Low-light experiment: The purpose of this evaluation is to assess the Computer Vision localization system at night, on board the Stena Flavia vessel. For the 2 trajectories we have used, the indoor spaces were only lit by the internal lighting of the vessel.

Description: For this experiment we have used the same trajectories (T01, and T02) with Section 7.2.4**Error! Reference source not found.** to provide a fair comparison between the remote and the field studies.

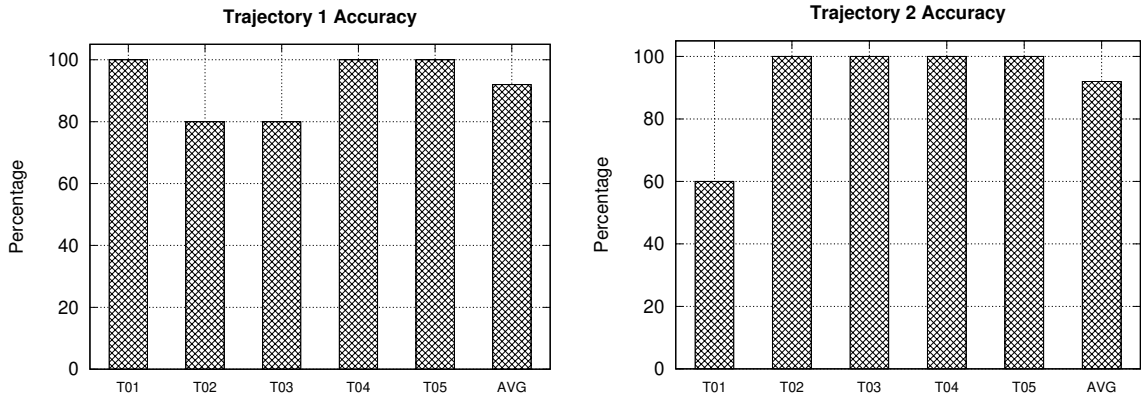


Figure 44: *Showing the Trajectory Accuracy results when following Trajectory 1 (left side) and Trajectory 2 (right side) on the field study. The average accuracy in both trajectories was 92%. The Trajectory 1 had varying performance when compared to the remote study, while the Trajectory 2 had the exact same results.*

Figure 44 shows the Trajectory Accuracy results for the 2 trajectories on the field study. For Trajectory 1, the T02 and T03 success was slightly decreased to 80%, while the T04 and T05 had a perfect 100% localization success. Trajectory 2 was the same as with the remote study. The lighting conditions in this experiment did not play a significant role on the performance of the model. This was because adequate training was performed with footage that had included similar lightning conditions, i.e., no light from outside bleeding in, and relying only the vessel's indoor lights. The fluctuation in accuracy for the Trajectory 1 was expected and insignificant.

7.3.4 Localization Drill

During the second day on Stena Flavia, we have performed a drill to assess the SMAS system. ***Figure 45*** shows different stages of the drill, while in action. In total, 26 crew members took part in the drill. This included the fire patrol (also called the *runner*), two fire teams (of 4 members each), and both captains of the vessel. Four different devices have been used, 3 CAT S62 devices (for the patrol on deck 5 and each of the fire-teams, initially on deck 3), and a Samsung Galaxy S20 (on the bridge). An ad-hoc network connectivity was used, by setting up locally a Wi-Fi router. The local network had no connectivity to the Internet. However, this was not needed as the 4 devices as well the SMAS server were communicating internally on the vessel. The internal networks of the vessel could not have been used, as this required major configuration changes from the IT department of Stena. Once everything was set up, the bridge could monitor where the crew members were.

Scenario: The drill scenario was that one of the smoke detectors went off. Initially, the "runner" went to the spot of the smoke detector to confirm the existence of a fire. The selected location of the fire was on deck 5. The fire was then confirmed, and then the first fire-team was sent to the wrong place on deck 5. This was on purpose, for creating some artificial confusion for the drill. Then, the SMAS chat communication was initiated. The bridge has requested from the "runner" to issue a SMAS Alert. When this was done, the correct location was broadcasted to the bridge, as well to all crew members. Then, the first fire-team have moved to the correct location of the fire. This was followed by the mobilization of a second fire-team. Finally, the fire-teams have tackled the thread, and the bridge has requested from the "runner" to silence the alert.

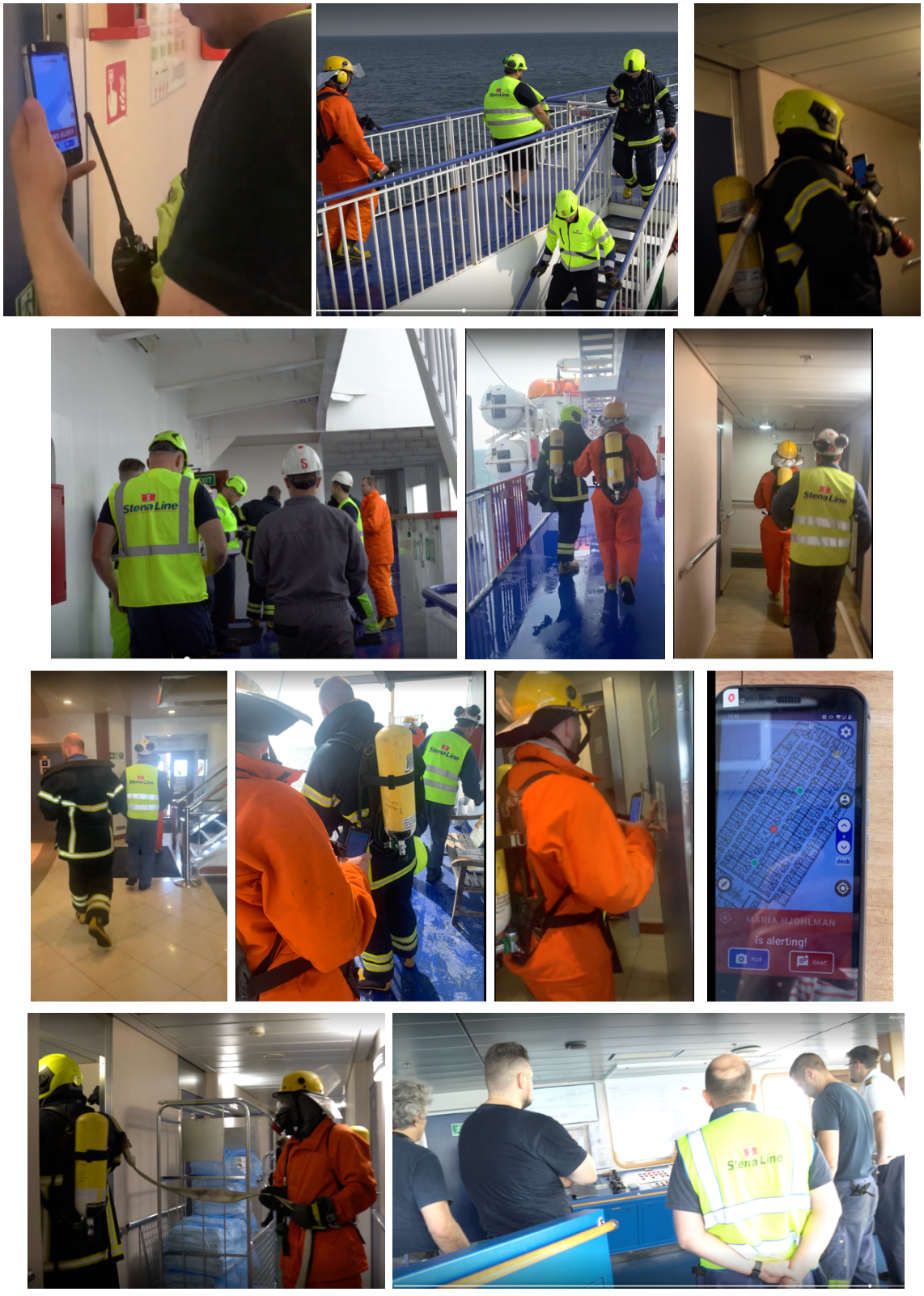


Figure 45: *The drill in action. In the first row: the runner (fire patrol) and identifies the fire while using SMAS in "Tracking Mode"; fire-team A in action; a fire-fighter from Team A interacting with SMAS. On the second row: fire-team B awaits instructions from the bridge; fire-team B on the move on the remaining pictures of this row. On the third row: fire-team B leaving from the restaurant area; fire-team B using SMAS and communicating with the bridge; fire-fighter in team B utilizing SMAS; the bridge inspecting both fire-teams' locations. In the fourth row: fire-team tackling the fire; the crew is back at the bridge, after the successful tackling of the incident and reports to the captain.*

7.3.5 Critical analysis, discussion, and Future ideas

Critical analysis:

- Some small signs were easily recognizable from longer distances, whereas some other bigger signs had troubled being recognized from shorter distances. This indicates (as discussed earlier) that additional training might be required to improve the model, capturing objects from different angles and different lightning conditions.
- Some areas in the logging process required repetitive work. For example, the latching points on car and cargo spaces (i.e., deck 3, 4, and 5), are spread over a wide area. For those points it may be faster in the future to refine the SMAS logger interface in a way to allow repetitive scene logs.
- Localization between tall trucks needed improvement. Relying on the latching points alone was not enough. The experimental IMU option has provided some autonomy, however, after some movement there was some drift (a well-known limitation of IMU-only based approaches). A future implementation could combine IMU with Object Recognition as well as QR or OCR codes.
- A battery optimization was implemented, which automatically disables the "Tracking Mode" (i.e., continuous localization) after 30 consecutive scans where no objects have been detected. However, the crew members have suggested that this value is very low, as the optimization was engaged when the crew was busy with other operations, like wearing their gear. Note that this is a configurable option, through the SMAS application settings.
- We have noticed that during daytime, in Deck 4 (where natural light is bleeding into an indoor space) and Deck 5 (on the open space), the direct sunlight is affecting the Computer Vision model.

Future improvements and ideas:

- Focusing on simpler objects (e.g., only considering signs) could significantly decrease logging time and the model's accuracy. This is because complex 3D objects have less distinguishable features when it comes to object recognition. An idea would be to place additional signs on the floors (i.e., in drencher zones in cargo spaces). This can provide a the 10m accuracy in areas of low object population, which is satisfactory for this project. Stena might be considering this option for their fleet, either with sticker signs or painted signs, following a consistent pattern and a glare-free material. Having an outline for each of those signs and being distinguishable from the background colors can help even further.
- Some QR code signs could be placed in locations where the patrol uses other equipment to check-in into different locations. This would automate this process and speedup the time required to patrol.

- Given that the previous 2 points are materialized, some more complicated objects can be removed from the model. This would simplify the training process.
- OCR can be extended to include cabin signs and cabin numbers.
- A wide-angle camera (or another setup with two cameras) could allow crew members to walk more freely in the indoor spaces. That camera setup could be mounted on a helmet or at chest level. During the drill, the crew had to point the camera towards objects, so they can be captured by the camera and subsequently be recognized by the system.
- Using an ASIC (application specific) chip, built for Deep Learning inference workloads (like the tensor chips) from Google, could allow up to 30x more recognitions when compared to the CAT S62 smartphones. This would allow significantly more recognitions, which in turn can improve localization accuracy.
- The localization could be extended to optionally log new objects that may be added later. This could be done while a regular patrol is being performed (i.e., self-adapting logging).
- By keeping the trajectories, the bridge can inspect whether the patrols have been properly performed and in an optimized fashion.
- An interesting future application that relies on indoor localization is to re-use the same Object Recognition model and the mapping of the vessel to provide navigation for autonomous vehicles (i.e., drones) that could scan the floor of the cargo decks and search for dangerous substances, detect overheated cars, or illegal passengers.

7.4 Impact Assessment and Cost Assessment

7.4.1 Impact Assessment

Equipping first responders with powerful mobile computing devices will allow them to increase their cyber-physical senses (i.e., multiple sensing devices like heat scanner in a tiny device), be connected (with the bridge and other personnel, discarding possibly outdated communication gear), be informed (e.g., carrying bulky manuals and maps in digital form) and location-aware (i.e., localization, navigation and tracking of mobile and static assets). These are all dimensions that will increase fire safety by the means of state-of-the-art information technology that has proven itself in everyday life scenarios and that is for the same reason also unobtrusive, with a low learning curve, adaptable through software and economically viable for massive deployment.

Our artifacts have been tested extensively both in the laboratory and the field through a remote evaluation study and an on-board study. We particularly modelled very carefully the indoor spaces of the Stena Flavia ro-pax vessel and created the following outcomes: (i) an interactive search engine for the indoor Points-of-Interest of the vessel that can dynamically be edited through the Anyplace Architect; (ii) a machine learning model coined LASHCO that correctly recognizes objects of interests on the Stena Flavia and other vessels of the Stena fleet with similar equipment; (iii) a novel “zero” localization pipeline founded on object fingerprints that can execute directly even on a smartphone device with excellent performance; (iv) a complete Smart Alert System that enables first responders to have location-aware interaction through a multimedia chat interface and an alerting interface; (v) an evaluation of the SMAS developments through a remote study with supplemented video traces but also a field study during a real drill.

In conclusion, we believe that our “zero” infrastructure localization technology can play an important role on ro-ro vessels in the complete identification-tracking-positioning spectrum, namely live fire detection and localization, live monitoring, and tactical support, monitoring of cargo on a map, quality control and optimization of cargo load and distribution on ro-ro vessels. Finally the device is very light and will not interrupt on the daily routines of the operator.

7.4.2 Cost Assessment

In order to assess the cost of using our system, we split the cost in installation costs, license costs and operation costs that we discuss separately.

Installation Costs

Smartphone with Flir Heat Scanner	500 Euros x 10 devices = 5000 Euros
PC Server on the Bridge	1000 Euros
(Estimated) Human Training ML Model per Vessel (Stena Flavia size)	25,000 Euros or 0 Euros in case of self-training.
(Estimated) Human Logging per Vessel (Stena Flavia size)	25,000 Euros or 0 Euros in case of self-logging

License Costs

Our solutions are developed with open and free technology having no barriers-of-entry and a low cost of operation and maintenance. Additionally, our system is open-source and can both be used for preliminary assessing the benefits of our SMAS system but also for extending the D06.6 outcomes it into a real production system in the future.

Operation and Support Costs

In order to run our localization system in a production environment as a service requires a careful business plan that we plan to develop in the future. For the time being, the system can be assessed free-of-charge by the IT department of vessel IT departments. The vessel operators can also contribute to the open-source project to allow the project grow in the future, as we already have experience with one of the largest rail transport manufacturers in Europe and worldwide [4].

8 Conclusion

Main author of the chapter: Demetris Zeinalipour, UCY

The aim of Task T06.10 therein, was the development and demonstration of smart alert of nearby first responders. Particularly, the research objective was to develop an innovative geo-positioning technology to allow more efficient first response to initial fires on ro-ro vessels. Besides the core geo-positioning technology, the aim was also to develop a ship indoor information system and an application-based platform of a ro-ro indoor navigation and indoor fire intelligence system.

Our proposed architecture comprises data, localization, and network layers, in a complete ***SMart Alert System (coined SMAS).*** SMAS is the application exploiting the Anyplace A4IoT localization architecture to enable localization, advanced patrolling and efficient first response. SMAS achieve its core localization workflow through three stages, namely: (i) Training, where vessel owners supply video recordings of vessel interior spaces and these videos are used to train a model to recognize objects of interest; (ii) Logging: where the model is used to link spatial coordinates (longitude, latitude and deck layer) with surrounding objects on a map interface; and (iii) Localization: where arbitrary surrounding

objects are resolved into spatial coordinates (longitude, latitude and deck layer). We achieve the above through a novel ranking algorithm we developed and tested, coined Surface.

We also carried out verification tests at the University of Cyprus and additionally prepared demos of components to be integrated in the future as part of the UCYCO machine learning model. As a result, through D06.6 we carried out a complete development and evaluation of our infrastructure-free localization method running on commodity smartphone devices by nearby responders and also developed a complete Ship Indoor Information System for disseminating smart alerts.

We believe that our developments will have a long-lasting impact on the problem of infrastructure-less localization on vessels on ro-ro vessels, which effectively will have an impact in the complete identification-tracking-positioning spectrum, namely live fire detection and localization, live monitoring, and tactical support, monitoring of cargo on a map, quality control and optimization of cargo load and distribution on ro-ro vessels.

9 References

[1] "Indoor Quality-of-Position Visual Assessment using Crowdsourced Fingerprint Maps", Christos Laoudias, Artyom Nikitin, Panagiotis Karras, Moustafa Youssef, Demetrios Zeinalipour-Yazti, ACM Transactions on Spatial Algorithms and Systems (TSAS '21), Association for Computing Machinery, Vol. 7, Iss. 2, New York, NY, USA, 2021.

[2] "ASTRO: Reducing COVID-19 Exposure through Contact Prediction and Avoidance", Chrysovalantis Anastasiou, Constantinos Costa, Panos K. Chrysanthis, Cyrus Shahabi, and Demetrios Zeinalipour-Yazti, ACM Transactions on Spatial Algorithms and Systems (TSAS '22), Association for Computing Machinery, New York, NY, USA, 2022.

[3] "COVID-19 Mobile Contact Tracing Apps (MCTA): A Digital Vaccine or a Privacy Demolition?", Demetrios Zeinalipour-Yazti and Christophe Claramunt, Proceedings of the 21st IEEE International Conference on Mobile Data Management (MDM '20), IEEE Computer Society, pp. 1-4, doi: 10.1109/MDM48529.2020.00020, June 30 - July 3, 2020, Versailles, France, 2020.

[4] “The Anyplace 4.0 IoT Localization Architecture”, Paschalis Mpeis, Thierry Roussel, Manish Kumar, Constantinos Costa, Christos Laoudias, Denis Capot-Ray Demetrios Zeinalipour-Yazti, Proceedings of the 21st IEEE International Conference on Mobile

Data Management (MDM '20), IEEE Computer Society, pp. 218-225, doi: 10.1109/MDM48529.2020.00045, June 30 – July 3, 2020, Versailles, France, 2020.

[5] "A Geolocation and Smart Alert System for Nearby First Responders on Roll-on/Roll-off Vessels", Paschalis Mpeis, Jaime Bleye Vicario and Demetrios Zeinalipour-Yazti, ERCIM News 123, Special theme: Blue Growth (ERCIM News), Vol. 123, 2020, ISBN: ISSN: 0926-4981, 2020.

[6] "Internet-based Indoor Navigation Services", Demetrios Zeinalipour-Yazti, Christos Laoudias, Kyriakos Georgiou and Georgios Chatzimiloudis, IEEE Internet Computing (IC '17), IEEE Computer Society, Vol. 21, Iss. 4, pp. 54-63, Los Alamitos, CA, USA, 2017.

[7] "Wi-Fi Fingerprint-based Indoor Positioning: Recent Advances and Comparisons", S. He, S.-H.G. Chan, IEEE Communications Surveys & Tutorials, vol. 18, no. 1, doi: 10.1109/COMST.2015.2464084, pp. 466-490, 2016.

[8] "Location of things (lot): A review and taxonomy of sensors localization in iot infrastructure," R. C. Shit, S. Sharma, D. Puthal, and A. Y. Zomaya, IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2028–2061, 2018.

[9] "Iot data prefetching in indoor navigation SOAs, "A. Konstantinidis, P. Irakleous, Z. Georgiou, D. Zeinalipour-Yazti, and P. K. Chrysanthis, ACM Transactions on Internet Technology (TOIT), vol. 19, no. 1, pp. 1–21, 2018.

[10] "Distributed In-Memory Processing of All k Nearest Neighbor Queries", Georgios Chatzimilioudis, Constantinos Costa, Demetrios Zeinalipour-Yazti, Wang-Chien Lee and Evaggelia Pitoura, IEEE Transactions on Knowledge and Data Engineering (TKDE '16), Vol. 28, Iss. 4, pp. 925-938, 2016.

[11] "Rayzit: An Anonymous and Dynamic Crowd Messaging Architecture", Constantinos Costa, Chrysovalantis Anastasiou, Georgios Chatzimilioudis and Demetrios Zeinalipour-Yazti, Proceedings of the 3rd IEEE International Workshop on Mobile Data Management, Mining, and Computing on Social Networks, collocated with IEEE MDM '15 (Mobisocial '15), Vol. 2, pp. 98-103, Pittsburgh, PA, USA, 2015.

[12] "YOLOv4: Optimal Speed and Accuracy of Object Detection", Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao:. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13024-13033, doi: 10.1109/CVPR46437.2021.01283, 2021.

[13] "SMAS: A Smart Alert System for Localization and First Response to Fires on Ro-Ro Vessels", Paschalis Mpeis, Athina Hadjichristodoulou, Jaime Bleye Vicario and Demetrios Zeinalipour-Yazti, Proceedings of 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), Association for Computing Machinery, pp. 4, 27th June – 30th June 2022, Copenhagen, Denmark, DOI: https://doi.org/10.1145/3524860.3543282, 2022.

[14] "Zero Infrastructure Geolocation of Nearby First Responders on Ro-Ro Vessels", Paschalis Mpeis, Jaime Bleye Vicario and Demetrios Zeinalipour-Yazti, Proceedings of the International Conference on Computer Applications in Shipbuilding (ICCAS '22), The Royal Institution of Naval Architects (RINA, est. 1860), Yokohama, Japan, September 13-15, 2022, DOI: , 2022.

[15] "AnyplaceCV: Infrastructure-less Localization in Anyplace with Computer Vision", Paschalis Mpeis, Athina Hadjichristodoulou, Ioannis Ioannides, and Demetrios Zeinalipour-Yazti, The 23rd IEEE International Conference on Mobile Data Management (MDM '22), IEEE Press, pp. 290-291, June 6 - June 9, 2022, Paphos, Cyprus, DOI: 10.1109/MDM55031.2022.00062, 2022.

10 Indexes

10.1 Index of tables

Table 1: SMAS Testing Framework Example used for Unit Testing and Stress Testing 12
Table 2: HTTP Response Message from the SMAS Application Protocol Interface. 16
Table 3: SMAS app User Authentication Endpoints 17
Table 4: SMAS User Location Report and Get endpoints 20
Table 5: Example Fingerprint Localization Scenario 23
Table 6: SMAS Localization Endpoint 24
Table 7: SMAS Geo-Location Chat Send Subsystem API 28
Table 8: SMAS Geo-Location Chat Get Messages API 29
Table 9: Example of the SMAS Computer Vision (CV) Fingerprint 31
Table 10: Example of the SMAS CV Fingerprint Database 31
Table 11: List of Objects extended with OCR technology 35
Table 12: Uploading SMAS Logging Data to the SMAS Service 35
Table 13: Uploading SMAS Fingerprint Database Download 36
Table 14: Labels used to Train the Stena Flavia videos and generate a respective LASHCO model 43
Table 15: Multiset Subtraction used in the Surface Algorithm 54
Table 16: Decimal Precision of the WGS84 formatted location gives an improved method on clustering nearby Fingerprints. We particularly use 4 decimal places, i.e., an accuracy of about 11.112 meters 57
Table 17: The OBJECT_COUNT with WGS84 Clustering Implementation in SQL 57
Table 18: The Surface Global (SG) Algorithm Implementation in SQL 60
Table 19: The Surface Local (SL) Algorithm Implementation in SQL 61
Table 20: Video Files Used for the Remote Study 62
Table 21: OBJECT_FREQUENCY View of the Remote Study Logging Task. The figure shows objects ranked by how rare they were with the WGS84 11 meter clustering method. The below weights are taken into account in the localization ranking function 69
Table 22: Summarizing the 15 scenes that were used in the localization experiments using 6 different parameters. The first column (#) shows the Scene identifier (i.e., Sxx). The next two columns show the video footage that was used, as well a Short Description. The remaining 6 columns present parameters that differentiate the scenes between them. 72

10.2 Index of figures

Figure 1: *The SMAS Architecture* 10
Figure 2: *The SMAS Component Diagram* 10
Figure 3: *The SMAS app splash screen (left), user login window (center) and login configuration screen (right) exposing the available functionality to users that are not authenticated yet.* 15
Figure 4: *The SMAS app main screen showing the various functionalities available to the first responder.* 18
Figure 5: *The SMAS app showing the location of various first responders with different colors (blue: active cv location, orange: manual location, green: active location of other responder, gray: inactive location of other responder.* 20
Figure 6: *The SMAS app First Responder Location; requesting the Where-I-Am functionality enables the camera system in the background to obtain the latest user location; which is subsequently presented on the map along with the location of other first responders. Sharing*

the location of a user becomes seamless as a long-press is all it takes to share any location on the vessel on the chat channel or some other medium (e.g., email). 22
Figure 7: *The SMAS Chat system enables first responders to exchange instant messages in real time to share mission critical information to fellow responders and the bridge on incidents of concern (e.g., fire in the initial stages).* 25
Figure 8: *The SMAS Geo-Location Chat subsystem allows first responders to interchange location-based messages text messages with the assistance of a) voice recognition, b) multimedia attachments, and c-d) attachment zoom and magnification functionality.* 26
Figure 9: *The SMAS Geo-Location Chat features the following functionality: a) location share; b) preference selectors to define recipients (all users, same deck users, k-nearest and users within a bounding rectangle); and c) visual cues to allow a first responder know that a text message has been received.* 27
Figure 10: *The SMAS Logger subsystem enables the installers of the localization service to associate locations on a vessel (longitude, latitude and deck) with the surrounding objects. (a) The Logger is enabled through the main settings menu, where a user can switch between the main SMAS app and the SMAS Logger; (b) When clicking the Scan button the SMAS Computer Vision subsystem starts collecting surrounding objects for a user defined interval (a user can cancel or discard a collection if necessary); (c) The fingerprints are uploaded to the fingerprint database (in case of intermittent connectivity stored locally until the cloud layer is accessible); (d) a logger can instantly hit the "Where-am-I" button to see if how the collected signals improve the localization accuracy at the given location or whether the collection of additional scans is necessary.* 32
Figure 11: *The SMAS Logger guidelines and settings. A variety of options enable battery optimization, caching and performance. An installer can also enable debug messages and select between online/offline localization and the localization algorithm. These are important for mission critical scenarios and to expose the capabilities of SMAS with Zero infrastructure.* 33
Figure 12: *Carrying out OCR on selected surfaces in the SMAS Logger provides greater resolution on the type of object identified (e.g., in the example we know that we have the STB Pilot door bunker and not any door but there is also additional autocorrection potential of the wrong captured words through the AndroidX SpellCheckerService).* 34
Figure 13: *The Stena Flavia mapped on Anyplace Architect. This is a one-off process per vessel that takes approximately 30 minutes per deck. Modeling Vessel Interior spaces (corridors and POIs) allows us building a semantic indoor space used for search, localization, and navigation.* 39
Figure 14: *Modeling Vessel Interior spaces (corridors and POIs) allows us building a semantic indoor space used for search, localization, and navigation. The above indoor model is loaded on the localization engine to provide the map, search, localization, and navigation along with chat and heat scanning.* 40
Figure 15: *POIs can be edited to customize the information space and this information will be available to first responders over their smartphone.* 40
Figure 16: *The Anyplace Viewer is a search engine for the indoor model constructed in Architect. It has been adapted for vessels providing cross-deck search and navigation over a web browser or mobile application. This navigation map can also be useful to other crew members as it provides a handy all-in-one-map that can either be used* 41
Figure 17: *Showing the Anyplace integration interfaces of SMAS, namely the building selector.* 41

Figure 18: *(left) Intel CVAT Training Environment and Job Monitoring User Interface. (right) we show the various classes used for annotating the videos with useful semantics that are integrated in the machine learning models.* 43
Figure 19: *Example of annotating charging stations inside the CVAT environment (we finally opted to annotate such charger clusters individually rather as one object). Revisiting the annotation and refining the model is straightforward (e.g., the revised model can be shipped as part of the next software update).* 45
Figure 20: *Example of the exported Machine Learning Models in Tensorflow Lite format*..... 47
Figure 21: *SMAS Database Relational Schema (showing base tables and views) for lash.db (main database) and the sharded <user>.db of each user that allows coping with data ingestions without scalability issues.* 48
Figure 22: *SMAS Database – Example User Table* 49
Figure 23: *SMAS Database – Example Location Table*..... 50
Figure 24: *SMAS Database – Example Fingerprint (Database) Table*..... 50
Figure 25: *SMAS Database – Example Fingerprint Table*..... 52
Figure 26: *Multiset Subtraction with a real CV Example. If the left figure is the query (and the right the database), then the dissimilarity is 2 (specificEngineRoomSign not found, also doorBlueOCR found only once). If the other way around, the dissimilarity is 1 (signTruck not found)* 55
Figure 27: *SMAS Database – Example of OBJECT_COUNT and OBJECT_FREQUENCY Views* .. 57
Figure 28: *Example with Bounding Rectangle Filtering of Fingerprints in the Surface Local (SL) Algorithm*..... 58
Figure 29: *Collecting and Managing Fingerprints with the SMAS Logger. All results are stored on the SMAS service and used for subsequent localization requests.*..... 64
Figure 30: *The modelling of a vessel (for the indoor and outdoor spaces) does not require physical presence. To accomplish this remotely, we perform an initial pass over the footage so we can understand how the real word spaces (shown in the first 3 pictures) map to the vessel's deck plan (shown in the last picture).* 65
Figure 31: *Examples of strong fingerprints on different scenes in the ro-ro and PAX public space. Each of those combinations of objects are unique for the entire vessel. Therefore, the SMAS Logger is mapping those to physical coordinates (i.e., modeling process), so they can be used for uniquely identifying a user's location.*..... 66
Figure 32: *The left plot shows the time that was required to perform a remote logging per deck on Stena Flavia using footage from 14 videos, while the right plot shows the number of objects that were fingerprinted per deck.* 67
Figure 33: *Seeing the fingerprints mapped on the SMAS Logger interface helps the installation team understand where additional object collection is necessary. For example, the above shows that deck 5 clearly has sufficient logged objects to enable accurate localization.* 68
Figure 34: *This plot shows the localization success percentage for each of the 15 Scenes (labeled S01-S15), as well the average. In more than two thirds of the scenes we were able to achieve between 80%-100% accuracy, with an average of 80%, in a total of 150 localization attempts.* 73
Figure 35: *Showing the Trajectory 1, that was in Cargo Areas in Deck 3. The first image shows the 5 points that were followed, along with their order in the trajectory. The remaining images show the objects recognized at each point of the trajectory.* 74

Figure 36: *Showing the Trajectory 2, that was in Passenger Areas in Deck 5. The first image shows the 5 points that were followed, along with their order in the trajectory. The remaining images show the objects recognized at each point of the trajectory.* 75
Figure 37: *Showing the Trajectory Accuracy results when following Trajectory 1 (left side) and Trajectory 2 (right side). Each trajectory had 5 points and we have repeated the experiment 5 times. The average accuracy in both trajectories was an impressive 90%.* 76
Figure 38: The Surface localization algorithm scales linearly with the increase in the size of the Fingerprint Database (FDB). 77
Figure 39: *Comparing the latency (in ms) when different messaging methods were used. For each method, we have sent 1000 messages. The error bars show 95% confidence intervals.* ..... **Error! Bookmark not defined.**
Figure 40: *The left plot shows the time that was required to perform logging per deck, while the right plot shows the number of objects that were fingerprinted per deck. Both experiments were performed during the field study, on board the Stena Flavia vessel.* 80
Figure 41: *Offline localization in action on-board the Stena Flavia. From left to right, we show the scenes 15, 11, and 13.* 81
Figure 42: *This plot shows the localization success percentage of the 15 Scenes (labeled S01-S15; described in Section 7.2.3), as well the average. This experiment was performed on-board the Stena Flavia vessel. For around half of the scenes we were able to achieve between 80%-100% accuracy, with an average of around 70%.* 82
Figure 43: *Showing the battery consumption during the logging and localization experiments. The results are shown per deck for each method (i.e., logging and localization), and the bar represents battery percentage drop on the CAT S62 smartphone devices. In total the logging process caused a 36% drop while the localization process caused a 14% battery drop.* 83
Figure 44: *Showing the Trajectory Accuracy results when following Trajectory 1 (left side) and Trajectory 2 (right side) on the field study. The average accuracy in both trajectories was 92%. The Trajectory 1 had varying performance when compared to the remote study, while the Trajectory 2 had the exact same results.* 83
Figure 45: *The drill in action. In the first row: the runner (fire patrol) and identifies the fire while using SMAS in "Tracking Mode"; fire-team A in action; a fire-fighter from Team A interacting with SMAS. On the second row: fire-team B awaits instructions from the bridge; fire-team B on the move on the remaining pictures of this row. On the third row: fire-team B leaving from the restaurant area; fire-team B using SMAS and communicating with the bridge; fire-fighter in team B utilizing SMAS; the bridge inspecting both fire-teams' locations. In the fourth row: fire-team tackling the fire; the crew is back at the bridge, after the successful tackling of the incident and reports to the captain.* 85