ZERO INFRASTRUCTURE GEOLOCATION OF NEARBY FIRST RESPONDERS ON RO-RO VESSELS

P Mpeis, University of Cyprus, Cyprus
J Bleye Vicario, Centro Jovellanos, Spain
D Zeinalipour-Yazti, University of Cyprus, Cyprus

SUMMARY
The early suppression of fires on ro-ro vessels requires rapid fire identification as a fire of medium growth exponentially reaches 50kW after only 1 minute. Fire patrol members (e.g., able seamen) are asked to act as first responders in such fire incident cases. They do however lack the necessary digital technology for immediate localization, verification and coordination with the bridge and other first responders. Indoor localization requires dense referencing systems (such as Wi-Fi, UWB, Bluetooth antennas), but these technologies require expensive installations and maintenance. Also, Satellite-based indoor localization is obstructed by the bulky steel structures of vessels, so this doesn't work either. Within the LASH FIRE project, an H2020 funded project (Grant Agreement #814975) in which this publication is framed, research has been carried out to develop a ground-breaking localization technology that requires zero infrastructure using computer vision on commodity smartphone devices attached to the gear of first responders. The developed solution comprises of three steps: (i) Training, where vessel owners supply video recordings that are processed on a deep learning data center to produce an accurate computer vision machine learning model; (ii) Logging, where a mobile app allows referencing non-movable objects to the *(x,y,deck)* coordinates of a vessel; and (iii) Localization, where first responders localize on a digital map. Additionally, in case a sparse communication network is available, first responders can share their location, emergency messages and heat scan images with nearby first responders and the bridge. Our proposed technology is shown to be 80% and 90% accurate for localization and tracking scenarios, respectively, in a study we carried out with video traces from a real ro-ro vessel. The overall developed Smart Alert System (SMAS), streamlines the lengthy fire verification, coordination, and reaction process in the early stages of a fire, improving fire safety.
Keywords: localization, first responders, fire safety, early suppression

1. INTRODUCTION

The early suppression of fires on ro-ro vessels requires rapid fire identification as a fire of medium growth exponentially reaches 50kW after only 1 minute. Fire patrol members (e.g., able seamen) are asked to act as first responders in such fire incident cases. Even though a variety of effective technologies are currently in deployment (e.g., VHF/UHF communication, push alarms, checkpoint RFID readers, heat scanners, fixed telephony, and temperature dashboards with drencher knobs at the bridge), a unified digital tool for immediate local localization but also verification and coordination with the bridge and other first responders is not available.

LASH FIRE is an international EU-funded research project aiming to significantly reduce the risk of fires on board ro-ro ships. A particular focus in the project is the development and validation of smart technical solutions for quick first response and effective fighting of fires in their initial stage. One challenging task was the development of an innovative geo-positioning technology (i.e., longitude, latitude, and deck with area level accuracy). To this end, we develope and demonstrate a ground-breaking localization system that requires zero infrastructure, reducing costs and maintenance. Our solution uses static visual 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 process and can then be utilized by any first responder.

Our localization subsystem does not require any sort of communication infrastructure, as the localization function is executed on the smartphone device with offline data and is, as such, considered a zero-infrastructure solution.

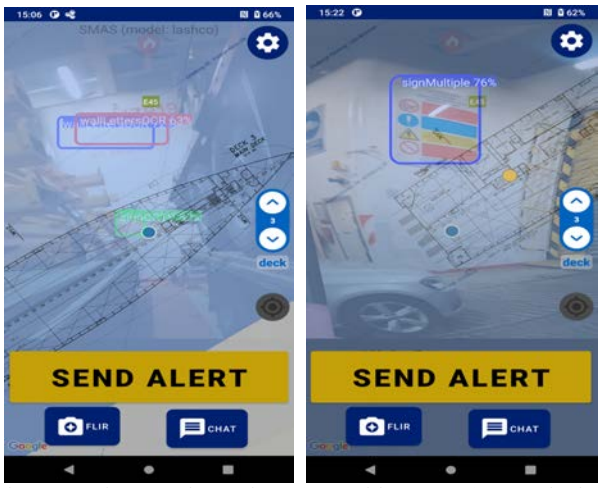

Figure 1: Fire patrol members (e.g., able seamen) are asked to act as first responders in fire incidents. We develop the *Smart Alert System (SMAS)* for smartphones that introduces a zero-infrastructure localization system using Computer Vision (CV).

Our proposed localization system relies on three stages:

- *Training:* vessel owners supply video recordings of vessel interior spaces that are consumed by a deep learning computing server that learns to recognize static vessel objects. This process is carried out once per vessel type (or vessel family, in case of vessels with similar objects);
- *Logging:* technology providers or vessel owners walk around the vessel collecting objects and associating them with locations (i.e., by long-pressing on a map), yielding a *Fingerprint DataBase (FDB)*. Logging is carried out once per unique vessel; and
- *Localization:* The first responders utilize a smartphone application that uses FDB to find their location when necessary.

In the context of this work, we present our zero-infrastructure localization algorithm, named *Surface,* whose aim is to provide area-level localization (i.e., within 10 meters). The name is inspired by the literal meaning of the word, namely our aim is to uncover *"the continuous set of objects that have length and breadth but no thickness"* and organize those objects together with their spatial coordinates *(x,y,deck)* in a database we name *Fingerprint Database (FDB)*. When a user (*u)* aims to localize, *u* queries in a transparent manner through a mobile app the set of objects currently stored in *FDB* retrieving the *(x,y,deck)* of the highest-ranked result.

Our Surface algorithm uses a combination of Global and Local pruning strategies, where the *Surface Global (SG)* algorithm is responsible to find a location from the complete *FDB*, while the *Surface Local (SL)* algorithm is responsible to find the next location from *FDB* based on the prior *(x,y,deck)*. The latter is used for tracking scenarios, where a continuous location over time is necessary. Realizing the SG and SL algorithms in a scalable and portable manner requires a high performance embedded spatial data management environment that we achieved using SQLite and innovative spatial algorithms.

Besides the core geo-positioning technology, our aim was also to develop a prototype vessel indoor location information system that will provide fire intelligence during patrol operations (for cases a sparse data communication network is available). For the above purpose, we developed a fully functional vessel communication software system, coined *Smart Alert System (SMAS),* which integrates our zero-infrastructure localization technology to a variety of subsystems (e.g., alert, nearest neighbor multimedia chat) allowing first responders to exchange messages and data (e.g., heat scans or images as shown in Figure 2). Although data communication networks might be widely available in ro-ro spaces, *dense deployment* of radio antennas necessary to provide accurate localization will remain an open problem because of installation and maintenance costs. Surface aims to fill this gap as it requires no infrastructure.

Additionally, by equipping first responders with powerful mobile computing devices has many benefits as it will allow them to increase their *cyber-physical senses* (i.e., multiple sensing devices, like heat scanner or measuring apps), *be informed* (e.g., carrying bulky manuals and maps in digital form), *be intelligent* (e.g., carrying deep learning neural networks that can recognize and track objects), *be location-aware* (i.e., localization, navigation and tracking of mobile and static assets) and *be connected* (with the bridge and other personnel, discarding possibly outdated communication gear). All these dimensions, packed in a tiny device, will increase fire safety to a new level by the means of state-of-the-art computer vision technology that has proven itself [7,8,9] and that is for the same reason also unobtrusive, with a low learning curve, adaptable through software and economically-viable for massive deployment.

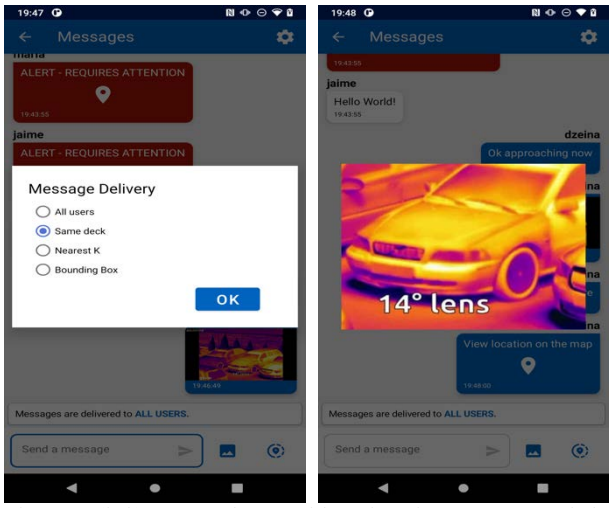

Figure 2: (left) SMAS also provides a location-aware vessel chat channel using the CV localization technology we develop; (right) integration with built-in heat scanners digitizes unnecessary gear and make the images available to nearby first responders.

To assess the correctness and usefulness of our propositions, we carried out an experimental study with extensive video footage from a real ro-ro vessel. We use these for a variety of tests and experiments in the laboratory showing a high accuracy of 80% and 90%, respectively, for localization and tracking. Our study shows that SMAS is an extremely promising technology that promotes situational awareness beyond the current state. There is no other technology that can provide infrastructure-free localization on a vessel nowadays. Our solutions are developed with open and free technology having no barriers-of-entry and a low cost of operation and maintenance.

2. RELATED WORK

The core technology for localization services is the type of hardware enabling the localization process [1,2,3,4,5]. This section surveys the current state.

2.1 NEW-INFRASTRUCTURE LOCALIZATION

These require the deployment of additional dedicated equipment, including proprietary transmitters, beacons, antennas, and cabling, for the provision of location signals. Industrial solutions in this field are termed *Real-Time Locating Systems (RTLS)* and current vendors specialize in specific markets for tracking assets using *Active or Passive RFID*, *Bluetooth Low Energy (BLE Smart) beacons* (e.g., Apple's iBeacon, Estimote.com context stickers), *Wi-Fi beacons* (e.g., Aeroscout.com Active RFID Tags and Kontakt.io cloud/BLE beacon) or *Ultra-Wide-Band (UWB)* chips (e.g., DecaWave.com UWB transceiver offering a 10cm localization accuracy) Assisted-GPS products (e.g., from Qualcomm) or multi-constellation solutions like CSR's SiRFusion chipset that employs information from GPS, Galileo, GLONASS and Compass satellites. These can improve availability in urban canyons and some indoor environments. However, to meet typical horizontal and vertical accuracy

requirements indoors, both solutions fuse satellite measurements with other complementary source of information including cellular and/or Wi-Fi and/or multiple inertial sensors [1].

On the other hand, a prominent academic effort is the *Epsilon* system by Microsoft Research, which relies on LEDs that flash in millisecond intervals, so that only a smartphone camera can pick up the pulses (e.g., one commercial effort that uses this idea is Bytelight.com). Subsequently, a smartphone user can be localized on the intersection of circles defined by those distances (i.e., *multi-lateration*). The emerging Li-Fi standard extends the idea of LEDs into communications. Another academic effort is the *ALPS* system by Carnegie Mellon University, which employs ultrasound audio signals captured by the smartphone's integrated microphone.

In theory, all solutions have the potential to achieve sub-meter level accuracy at high deployment densities. However, these solutions raise scalability issues in case of large indoor spaces. For example, Indoo.rs required 300 StickNFind beacons (i.e., estimated at 15,000 USD) to provide guidance to visually blind people at the 60,000 m2 Terminal 2 of the San Francisco Intl. Airport, USA. Additionally, there are costs associated with maintaining the batteries of beacons, tuning their signal levels, transmission frequencies as well as interference issues.

2.2 EXISTING-INFRASTRUCTURE LOCALIZATION

This category includes systems that use location-dependent measurements from existing wireless communication infrastructure, such as Wi-Fi access points and cellular base stations. Field tests revealed that these can attain localization accuracy that is comparable or even better than costly new infrastructure-based systems [2].

In this context, our team has over the years developed the Anyplace Indoor Information System [1,2,3], which is a Wi-Fi fingerprint localization system that won many awards for its utility and accuracy. Anyplace has traditionally only focused on Wi-Fi localization but will also support CV localization after this work given the fact that dense Wi-Fi installations are not frequently available on ro-ro vessel and other scenarios. In Figure 3, we show how the indoor model of a real ro-ro vessel used in this study was mapped using our Anyplace Architect tool (https://anyplace.cs.ucy.ac.cy).

In this category we can also classify a special type of RTLS that emerge by enterprise wireless LAN vendors. Companies such as Cisco, Ericsson, Ekahau.com, Arubanetworks.com and Aerohive.com allow enterprises to manage their enterprise WLAN networks but also offer location tracking extensions to their services. Also, Wi-Fi routers with proprietary operating systems (e.g., OpenWrt.org) can be configured for these scenarios to acquire the signal intensity of users and localize them.

Figure 3: (top) Mapping the ro-ro vessel in our evaluation using our Anyplace Architect tool; (bottom) Obtaining navigation instructions and information search in Anyplace Viewer.

2.3 ZERO-INFRASTRUCTURE LOCALIZATION

These are used in environments where there is no localization infrastructure available. In this category we find Magnetic localization systems (e.g., IndoorAtlas), which take advantage of the magnetic field anomalies typical of indoor settings by using them as distinctive place recognition signatures. This sort of technology offers accuracy without any hardware requirements and a relatively low total cost of ownership. On the other hand, these sorts of signatures work only for stationary steel structures in specific coordinates but not for vessels that are always moving to different locations. As such, magnetic localization is not applicable when it comes to mobile indoor spaces like vessels.

Infrastructure-free localization often also refers to solely IMU-based approaches that have been extensively studied in the context of Pedestrian Dead Reckoning (PDR) systems. Particularly, sensory data reported by *Inertial Measurement Units (IMU)*, including accelerometers, gyroscopes, and digital compasses (e.g., CyweeKIOS or WiFiSLAM) [1]. Such sensor modules are either integrated into modern consumer electronics, or attached externally on the human body, i.e., head, back, waist or foot mounted while processing occurs by low-power co-processor (e.g., motion coprocessors on iPhones). IMU solutions can be used to provide relative location of a mobile but is known to suffer from drifting (i.e., even the slightest localization or orientation error builds up over

short future windows yielding high location errors). As such, IMU solutions are known to require an infrastructure to correct the location signal over time. In Anyplace we traditionally utilize Wi-Fi to correct the IMU signal. Likewise in this work, we will consider in the future a hybrid tracking algorithm by the means of fusing the CV signals to the IMU signals, to cope with the drifting problem of IMU.

The SMAS zero-infrastructure localization system developed in this work, proposes the Surface algorithm, which exploits novel data management ranking algorithms with deep learning models trained on ro-ro vessel footage to carry out accurate zero-infrastructure indoor localization with smartphones.

3. SURFACE ALGORITHM

3.1 OVERVIEW

In this section we describe our Surface algorithm, explaining its rational through various examples. Our Surface algorithm uses a combination of Global and Local pruning strategies, where the *Surface Global (SG)* algorithm is responsible to find a location from the complete *FDB*; while the *Surface Local (SL)* algorithm is responsible to find the next location from *FDB* based on the prior *(x,y,deck)*. These algorithms deploy sophisticated object ranking functions founded on the concepts enumerated below and that are implemented with the expressive power of the *Standard Query Language (SQL):*

- *Multiset Subtraction (SG & SL algorithms),* where we scan through the *Fingerprint Database (FDB)* and identify the (*x,y,dec*k) locations that have their set of objects more closely to the query object-set.
- *Global Partitioned Frequency Counting (SG & SL algorithms)*, which captures the frequency a given object has after the FDB creation stage (i.e., after logging). This way, the Surface algorithm knows which objects are of high importance and takes those objects into the ranking process of candidates. The objective is to return the (*x,y,deck*) triples 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 SL 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 yielding no results, the global localization algorithm alleviates the problem and finds temporarily the best result until a better estimate can be made.

3.2 DATA PROCESSING CONCEPTS

In this section we explain the underlying concepts of the Surface algorithm in further detail.

3.2 (a) Multi-set 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.

The below example shows the basic object subtraction operation that takes place in Surface.

Example Multiset Subtraction (Query – FDB Record):

- {drencher, charger} – {drencher, charger} = ∅
  Dissimilarity = 0
- {drencher, charger} – {drencher, charger, door} = ∅
  Dissimilarity = 0
- {drencher, charger} – {drencher} = {charger}
  Dissimilarity = 1
- {door, door, door} – {door} = {door, door}
  Dissimilarity = 2
- {door} – {door, door, door} = ∅
  Dissimilarity = 0

3.2 (b) 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 stage. For an object, whether unique or frequent, we might have a dozen of fingerprints depending on the coverage effort during logging and the physical space characteristics. Effectively, 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 higher accuracy localization.

To alleviate this problem, we decided to carry out a spatial hashing of objects to locations and create the so-called *OBJECT_COUNT* view. This allows identifying how unique an object is in relation to its location. For example, if a *signGather* object appears in three locations then this will have a frequency counter of 3, otherwise if these three locations map to the same location the frequency counter will be 1. This also simplifies logging, as we can collect multiple fingerprints and retain stable frequency counters.

In order to normalize the importance of OBJECT_COUNT objects we average over the total number of objects yielding the OBJECT_FREQUENCY table. This was implemented as a materialized table, as opposed to view, to avoid computing the view for each query (i.e., for performance reasons).

3.2 (c) Spatial Partitioning of Fingerprints

Even though the original OBJECT_COUNT idea clusters fingerprints on (*oid,x,y,deck*), where *oid* is the unique object identifier, one problem is that the fingerprints will rarely be precisely on the same (*x,y,deck*) because the location we use in the logging process has very high precision (e.g., we use Google Maps that uses WGS84 with 1 meter accuracy.)

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). With 4 decimal places of precision, we get a clustering of ~11 meters. The above clustering will be exploited in both the SG and the SL algorithms. In Table 2 we present the OBJECT_FREQUENCY table for the top-5 (most rare) and lower-5 (most common) objects.

3.2 (d) 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 object-sets to irrelevant locations on the vessel in tracking scenarios (i.e., where the localization task takes place every few seconds or after the inertial sensor triggers a re-computation.)

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 decks, 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 decks away), we automatically run the global ranking SG algorithm that will find the closest match through the complete vessel.

In order to visualize the Bounding Rectangle, consider the example in Figure 4. Here we show the FDB 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 and usually also faster.

3.3 LOCALIZATION ALGORITHMS

3.3 (a) The Surface Global (SG) Algorithm

The SG algorithm is an algorithm that can be executed without any prior localization state. SG will search through the fingerprint database FDB to find the closest match that can be in any of the vessel decks. SG deploys multiset subtraction, global partitioned frequency counting and spatial partitioning of fingerprints to rank and return the closest fingerprint to the localization query. Given that SG is a less restrictive case of the Surface Local (SL) Algorithm, we present next, we will defer its detailed presentation for Section 3.3 (c) where we present both algorithms in a more elaborate fashion.

3.3 (b) The Surface Local (SL) Algorithm

This algorithm requires the prior location of a user (*prevX, prevY and prevDeck*) to find the most relevant fingerprints from FDB using the concept of Bounding Box Filtering of Fingerprints. SL searches for the closest dissimilar objects in a bounding box that is system-defined (e.g., 100m and not more than +/- 1 decks). Figure 4 presents an example: We observe that the bounding rectangle prunes the object space to a small subset of results, yielding a sub-ranking of results within this box.

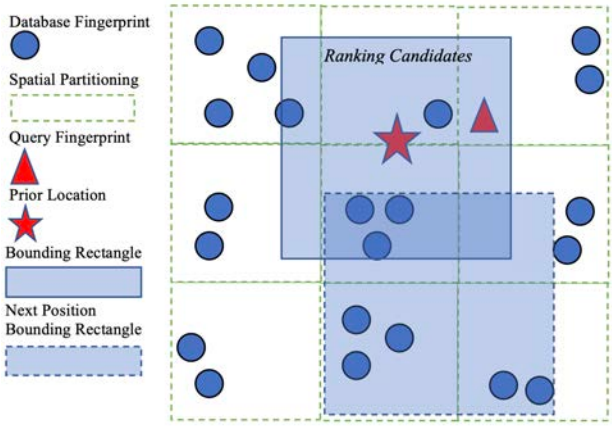

Figure 4: Example with Bounding Rectangle Filtering of Fingerprints in the Surface Local (SL) Algorithm.

The SL algorithm is particularly suited for tracking scenarios where fingerprints need to be dynamically re-ranked based on where a user currently is. Of course, such a sub-ranking might not always yield a result. As such, the SL algorithm will invoke the SG algorithm (global ranking) as fallback case in case it yields an ∅ set.

3.3 (c) Surface Implementation in SQLite

Figure 5 presents the implementation of the Surface algorithm in SQLite, which is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. 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. For brevity we will explain the common components of SL and SG together and finally focus on the additional restrictive aspect of the SL algorithm.

```
1  SELECT F.fid, F.X, F.Y, F.deck,
2         ABS(x - "$prevX") as xDiff, ABS(y - "$prevY") as yDiff, ABS(deck - "$prevDeck") as deckDiff,
3         (SELECT IFNULL(COUNT(*),0) FROM
4    ( -- Ranking Criterion A: Multi-set Subtraction as
5    SELECT ROW_NUMBER() OVER (PARTITION BY FLT.oid) AS RowNum, FLT.oid
6    FROM FINGERPRINT_LOCALIZE_TEMP FLT WHERE FLT.uid="$uid"
7    EXCEPT
8    SELECT ROW_NUMBER() OVER (PARTITION BY FO.oid) AS RowNum, FO.oid
9    FROM FINGERPRINT_OBJECT FO WHERE FO.fid=F.fid
10   )
11 ) AS dissimilarity, (SELECT IFNULL(AVG(weight),1) FROM
12   ( -- Ranking Criterion B: Multi-set Subtraction on Global Partitioned Frequency Counting with Spatial Partitioning of Fingerprints
13   SELECT ROW_NUMBER() OVER (PARTITION BY FLT.oid) AS RowNum, FLT.oid, OF.weight as weight
14   FROM FINGERPRINT_LOCALIZE_TEMP FLT, OBJECT_FREQUENCY OF
15   WHERE  FLT.oid = OF.oid and FLT.uid="$uid"
16   EXCEPT
17   SELECT ROW_NUMBER() OVER (PARTITION BY FO.oid) AS RowNum, FO.oid, OF.weight as weight
18   FROM FINGERPRINT_OBJECT FO,  OBJECT_FREQUENCY OF
19   WHERE  FO.oid = OF.oid and FO.fid=F.fid
20   )
21 ) AS weight
22 FROM FINGERPRINT F
23 -- Bounding Rectangle Filtering of Fingerprints
24 WHERE F.modelid="$modelid" and F.buid="$buid" and
25       (x between "$prevX" - "$smas_db_location_bound_meters" and "$prevX" + "$smas_db_location_bound_meters") and
26       (y between "$prevY" - "$smas_db_location_bound_meters" and "$prevY" + "$smas_db_location_bound_meters")
27       and (deck between "$prevDeck" - 1 and "$prevDeck" + 1)
28 -- keep only x,y,deck rounded by 10m bounding box
29 GROUP BY ROUND(F.X,"$smas_db_location_bound_rounding"), ROUND(F.Y,"$smas_db_location_bound_rounding"), F.deck
30 -- include results that have at least 1 common object
31 HAVING dissimilarity < (SELECT COUNT(*) from FINGERPRINT_LOCALIZE_TEMP FLT WHERE FLT.uid="$uid")
32 -- rank by dissimilarity, then OBJECT_FREQUENCY weight than by distance from prior location
33 ORDER BY dissimilarity, weight, ABS(x - "$prevX") + ABS(y - "$prevY") + ABS(deck - "$prevDeck") ASC
34 -- return highest ranked result only
35 LIMIT 1;
```

Figure 5: Implementation of the Surface Local (SL) Algorithm in SQLite as part of the SMAS backend.

The common parts of both algorithms are the following: (i) the computation of a dissimilarity score using the notation of multiset subtraction. This is achieved as part of the pre-last predicate in the outermost SELECT block (lines 4-10); (ii) the computation of the frequency ranking given higher ranking precedence to rare objects. This is achieved as part of the last predicate in the outermost SELECT block (i.e., lines 12-20). The clause IFNULL(AVG(weight),1) aims to penalize objects in the ranking process for objects that are part of the Query but not the Database. For all cases, given that ranking is a global process, which might yield a large result set, we filter the results set on two predicates:

- **Partitioning:** We exploit the spatial partitioning of fingerprints that we discussed in object frequency discussion (line 29).
- **Overlap:** We require at least one object overlap between the Query and the FDB (line 31).
- **Sorting:** We sort the results by importance (line 33).
- **Top-1:** We return the highest-ranked result (line 35).

Contrary to the SG algorithm, the SL algorithm computes the following two additional concepts:

- **Bounding Rectangle Filtering of Fingerprints**: that filters results using the bounding box we described earlier (line 24).
- **Query Deviation**: that provides the deviation of the query from the returned result (line 2).

3.3 (d) Discussion

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). 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 relational algebra upon which the complete relational data management field is founded and succesful.
- *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 in the algorithm development stage and applying respective remedies if needed (e.g., indices, view materialization). I/O 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.

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).

4 EXPERIMENTAL EVALUATION

To assess the correctness and performance of the SMAS architecture and implementation, we carried out 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 (i.e., in vitro).

4.1 METHODOLOGY

Dataset: We use 14 videos captured by a commercial ro-ro operator as part of the LASH FIRE project with a total size of 11GB (i.e., approximately 800MB per file). The footage is from one specific ro-ro vessel that we name *VESSEL* for the remainder of this work. The dataset 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 research has been mainly the Closed ro-ro, Open ro-ro and Weather deck, which are the decks 3 and 4 of the given vessel, 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. Results from deploying our system on the real vessel might be presented in the future as we plan to carry out an on-board study of SMAS very soon.

Table 1: VESSEL Video traces used for training and localization.

Video ID	Size (MB)	Length (min)	Deck	Area	Annotation
V01	637.88	8.51	3,4	Closed ro-ro	ro-ro, elevator, stairs
V02	640.88	8.51	4	Closed ro-ro, Open ro-ro	Empty, Loading
V03	1698.15	3.56	4	Closed ro-ro	Loading
V04	638.80	8.51	4	Closed ro-ro	Loading
V05	643.77	8.51	4	Closed ro-ro	Loading
V06	644.05	8.51	4	Closed ro-ro	Loading
V07	385.17	5.12	4	Closed ro-ro	Empty
V08	50.79	0.42	5	PAX public	Open Deck, Reception, ASSY A
V09	546.31	7.3	5	PAX public	ASSY A, Restaurant, Open Deck, Reception, Passenger Deck
V10	129.56	1.44	5	PAX public	ASSY B, Open Deck
V11	476.28	6.29	6,5	PAX public, Cabin	cabin, stairs, restaurant, ASSY A loop
V12	649.69	8.52	5,6	PAX public	restaurant, reception, open-deck, stairs
V13	3521.50	8.09	3	Closed ro-ro	cars, trailers, containers
V14	639.02	8.51	4	Closed ro-ro, Open ro-ro	containers, cars

CV Model Training: Computer Vision Annotation Tool (CVAT) 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. To execute CVAT in our environment we use a dedicated deep learning server, namely an HP DL380 Gen10 with 80 logical processors and a powerful NVIDIA V100 card. 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. The videos were annotated by a team of 5 persons over a period of 4 weeks with several iterations, to refine the quality of the constructed LASHCO neural network model we built for VESSEL (24MB with approximately 100 classes). Our builds were initially encoded in YOLO [6]. One problem is that the YOLO models are not sufficiently efficient on smartphones, as such, we export the models in Tensorflow Lite [8], which is a mobile library for deploying models on mobile, microcontrollers and other edge devices. 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 an impact on the performance of the model with no great loss in accuracy.

Localization Hardware: We use a standard led monitor that shows the video files playing with the VLC tool. For the logging and localization tasks, we use the Caterpillar S62 smartphone, which was selected as the experimental apparatus for its rugged properties, battery lifetime and embedded heat camera. Particularly, the device 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 camera sounds very specific, the system works well with any smartphone having a capable camera.

Tuning Remote Logging: The video player 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. We observed this in the laboratory with both UCYCO, which was trained by us, and COCO [9] that comes pre-trained. As such, the logging and localization accuracy provides a worst-case bound on the accuracy we will obtain in a live environment. We aim to validate this claim in the on-board study where logging and localization will happen directly through the camera-system of our smartphone.

4.2 LOGGING TIME EVALUATION

The purpose of this evaluation was to observe the effort necessary to collect object-sets through the SMAS Logger, which subsequently will enable zero infrastructure localization on a vessel. For this series, our evaluation metric is expressed mainly in logging time, but we do also measure the number of objects that were collected per deck. Figure 6 present some example fingerprints in closed ro-ro, PAX public and shows how these are marked on a map (that subsequently registers the reading in FDB). We include fingerprints with varying light intensities as well as camera visibility obstructions (e.g., corridors between trucks).

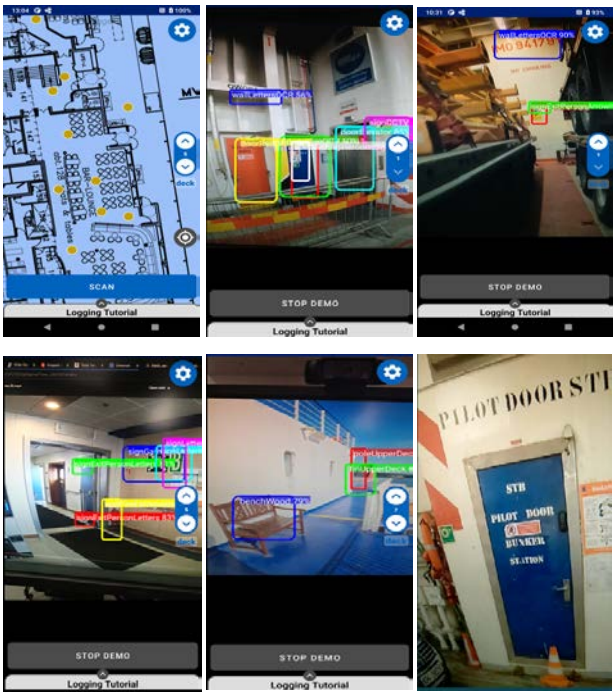

Figure 6: CV fingerprints collected from VESSEL and entered to the Fingerprint Database (FDB) through a map interface and OCR example shown last.

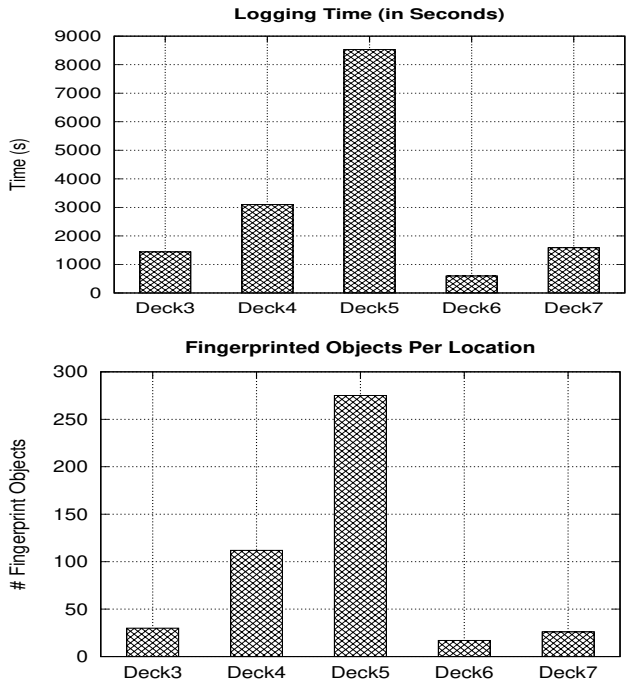

Figure 7: (top) Time necessary to perform the logging task per deck on VESSEL using footage from 14 videos; (right) Fingerprint Objects collected per deck.

Results: In Figure 7 (top), we show the logging time required to model the VESSEL. 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, amongst others. Deck 4 required about 10 mins, while Deck 3 and Deck 7 required less than 30 minutes. Deck 6 did not have a lengthy capturing and required just 10 minutes. For each spot it required roughly 3 mins. Given that we cluster close-by fingerprints in the OBJECT_FREQUENCY view, collecting many fingerprints per location is good.

In Figure 7 (bottom), we visualize the fingerprint objects that were captured during the modeling of VESSEL. In total about 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.

In Table 2 we present the top-5 and lower-5 objects identified in our logging task as part of the OBJECT_FREQUENCY view. The given view ranks objects by rarity using by using our WGS84 11-meter clustering method. These weights determine the ranking of localization results after the dissimilarity criterion. Note that all objects with the OCR are passed through Google's ML kit on-device *Optical Character Recognition (OCR)* library (available in multiple languages), which can refine an object recognition. For example, for Figure 6 (last) it reads *"STB PILOT DOOR BUNKER TTION"*, missing only the "station". OCR will be used further in the future.

Table 2: OBJECT_FREQUENCY View after Logging

Object ID (oid)	Object Description	Weight (rare to frequent)
98	signBathroomDisabled	0.0052631
95	signFireSafetyLever	0.0052631
93	wallSingleDigitOCR	0.0052631
92	specificTankDeck	0.0052631
91	signNumberOCR	0.0052631
…	…	…
49	wallPatternRedWhite	0.0315789
4	signExitPersonLetters	0.0315789
69	signLetterOCR	0.0368421
43	doorBlue	0.0368421
52	wallFireExtinguisherEmbedded	0.0473684

4.3 LOCALIZATION EVALUATION

The purpose of this evaluation was to remotely assess the CV localization system when localizing without any prior knowledge of the user's location (e.g., for the first time). Particularly, we test the effectiveness of the SG algorithm. Our aim is to observe whether Surface can distinguish in each of the 15 scenes the correct location within 10 meters (i.e., room or area-level accuracy).

Metric: We use *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, an expert user judges the closeness of the result. Specifically, the expert user judges that the localization query (i.e., $Result_i$) is successful

(i.e., $Result_i = Correct_i$), when the user localizes within 10 meters of the ground truth (i.e., $Correct_i$). The above is repeated *n* times (in our experiments n=10). Below we provide the definition of our accuracy using a set-theoretic notation, where Result indicates the returned location and Correct location the ground truth.

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

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 our project focused on the former areas. Table 3 lists the 7 different dimensions 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, trucks, passengers, or mixed), the density of those objects in the scene, and finally whether the camera that captured the footage was moving or not.

Table 3: Scenes used in SG Localization Experiment

#	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

The first scene, (i.e., S01) the user was in front of a lift in Deck 3, with many objects recognized. In S02, the user was standing in front of the Cargo Office in Deck 3, but only a few objects were recognized. In S03, the user was in front of walls that are surrounding a funnel, recognizing a few labels on that wall. In S04, the user was walking between trucks, 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 trucks or cars, recognizing some labels and doors on the outer wall of the vessel. In S07, the user was walking in open ro-ro space recognizing some labels and doors on the wall. In S09, the user was walking between trucks 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 S14, 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. Quite importantly, V13 was not used for building our ML model, as such, was a good test of our constructed model.

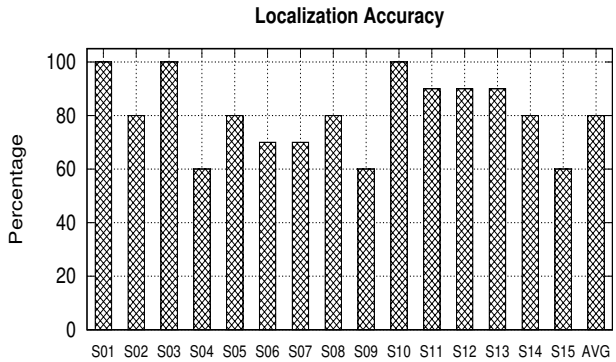

Figure 8: Localization Accuracy for the 15 scenes of Table 3 as well as the average (last column) showing that Surface exposes an 80% accuracy on average.

Results: Figure 8 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. 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 trucks on both sides and was not able to recognize a necessary number of objects before attempting to localize (for such scenarios the SL algorithm will be beneficial). There is a similar scenario in S09, with the only difference that the user was walking in a lane that had trucks 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 the remote execution of this experiment), as well the naturally low number of objects in those areas. We have scheduled an onboard study on a ro-ro vessel to verify these claims and will report new results in the future.

4.4 TRACKING EVALUATION

The purpose of this evaluation is to remotely assess the Computer Vision localization system when localizing on a particular trajectory. Particularly, we test the effectiveness of the SL algorithm in a tracking scenario (without IMU and without a particle filter fusion). We have used two different trajectories. We used footage from two different videos, namely V13 (closed ro-ro) and V09 (Public PAX), following two different trajectories playing at normal speed. Each trajectory had 5 points, and for each point we performed the localization 5 times. Figure 9 shows visually Trajectory 1 for V13 (closed ro-ro).

Figure 9: Trajectory 1 (with V13) used in Tracking Experiment

Figure 10 shows the Trajectory Accuracy results for the two 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. Capturing in a live scenario would have presumably corrected this issue and we will verify this claim in the planned onboard study.

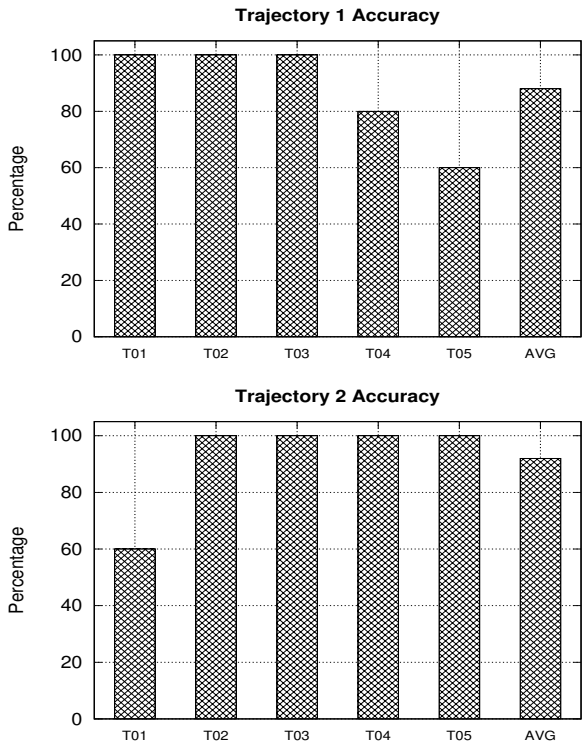

Figure 10: (top) Trajectory 1 with V13 and (bottom) Trajectory 2 (with V09); in the Tracking Experiment. The average accuracy in both trajectories was 90%.

4.5 SCALABILITY EVALUATION

The purpose of this 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 11 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.

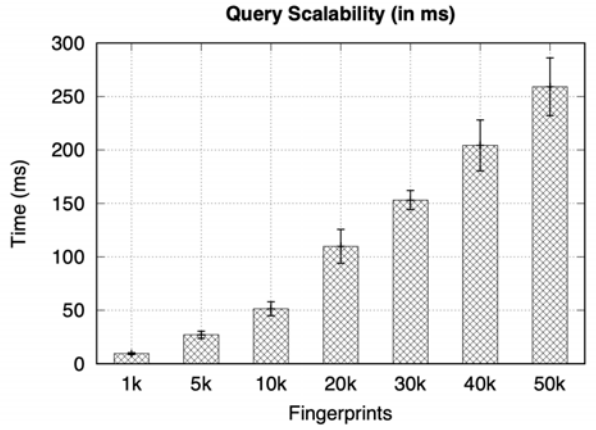

Figure 11: The Surface localization algorithm scales linearly with the increase in the size of the Fingerprint Database (FDB).

6. CONCLUSIONS

In this paper we present a ground-breaking localization technology that requires zero infrastructure using computer vision on commodity smartphone devices attached to the gear of first responders. Our solution deploys a data-driven localization algorithm, coined Surface, which operates with global and local optimizations. Surface has been found to be 80% and 90% accurate for localization and tracking scenarios, respectively, in a study we carried out with data from a real ro-ro vessel. The developed Smart Alert System (SMAS) streamlines the lengthy fire verification, coordination, and reaction process in the early stages of a fire, improving fire safety. Our developments are expected to have a long-lasting impact on the problem of zero-infrastructure localization 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, quality control and optimization of cargo load and distribution.

7. ACKNOWLEDGEMENTS

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

8. REFERENCES

1. Demetrios Zeinalipour-Yazti, Christos Laoudias, Kyriakos Georgiou and Georgios Chatzimiloudis, "Internet-based Indoor Navigation Services", IEEE Internet Computing (IC '17), IEEE Computer Society, Vol. 21, Iss. 4, pp. 54-63, Los Alamitos, CA, USA, 2017.
2. Paschalis Mpeis, Thierry Roussel, Manish Kumar, Constantinos Costa, Christos Laoudias, Denis Capot-Ray Demetrios Zeinalipour-Yazti, "The Anyplace 4.0 IoT Localization Architecture", 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.
3. Christos Laoudias, Artyom Nikitin, Panagiotis Karras, Moustafa Youssef, Demetrios Zeinalipour-Yazti, "Indoor Quality-of-Position Visual Assessment using Crowdsourced Fingerprint Maps", ACM Transactions on Spatial Algorithms and Systems (TSAS '21), Association for Computing Machinery, Vol. 7, Iss. 2, New York, NY, USA, 2021.
4. Rathin C. Shit, Suraj Sharma, Deepak Puthal, and Albert Y. Zomaya, "Location of things (lot): A review and taxonomy of sensors localization in iot infrastructure", IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2028–2061, 2018.
5. Andreas Konstantinidis, Panagiotis Irakleous, Zacharias Georgiou, Demetrios Zeinalipour-Yazti, and Panos K. Chrysanthis, "Iot data prefetching in indoor navigation SOAs", ACM Transactions on Internet Technology, vol. 19, no. 1, pp. 1–21, 2018.
6. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13024-13033, doi: 10.1109/CVPR46437.2021.01283, 2021.
7. Tesla Autopilot. Future of Driving https://www.tesla.com/autopilot
8. Tensorflow Lite. Deploy machine learning models on mobile and edge devices. https://www.tensorflow.org/lite
9. COCO: Common Objects in Context https://cocodataset.org

9. AUTHORS BIOGRAPHY

Paschalis Mpeis is a Postdoc Researcher at the Department of Computer Science at the University of Cyprus and holds a PhD from the University of Edinburgh, UK. His research includes mobile systems. He is the lead developer of the open source SMAS app.

Jaime Bleye Vicario is a Seaman and Nautical Engineer. Is the Head of the firefighting area at Jovellanos Training Centre, Salvamento Maritimo (SAS). He has specialized in marine firefighting operations, publishing a "Manual about interventions of land firefighters on board vessels" edited by the Port Authority in Spain. He has working in previous EU Projects and now is leading WP06 "Manual interventions" in the LASH FIRE project.

Demetrios Zeinalipour-Yazti is an Associate Professor of Computer Science at the University of Cyprus and holds a PhD from the University of California – Riverside, USA. His research includes data management in systems and networks, particularly mobile data management. He is an ACM Distinguished Speaker, a Senior Member of ACM, a Senior Member of IEEE, and a Member of USENIX, a Humboldt Fellow (Germany) and a Marie-Curie Fellow (EU). He is the technical leader of the SMAS system and the lead develop of the SMAS backend.