Automated Product Localization Through Mobile Data Analysis

Magnus Karsten Oplenskedal[1,3], Amir Taherkordi[1,2,3] and Peter Herrmann[1]
[1]Norwegian University of Science and Technology (NTNU), Trondheim, Norway
[2]University of Oslo, Norway
[3]Forkbeard Technologies, Oslo, Norway
{magnukop, amirhost, herrmann}@ntnu.no

***Abstract*—Recent developments in the field of indoor Real-Time Locating Systems (RTLS) using mobile devices stimulate decision support for users. For instance, smartphone-based navigation in shops can enable location-aware recommendations of certain products to customers. An impeding factor to realize such systems is that they need the exact position of products. Existing product localization solutions, however, are based on tagging or manual location registering which tend to be quite costly and laborious. In this paper, we propose an *automated product localization* approach solving this problem. Our system infers the location of products based on the results of accumulating two sets of customer data, i.e., the locations at which the customers stop for picking up items as well as the list of the items, they purchase. These two data sets are accumulated for a large number of users, making it possible to build correct mappings between the products and their positions. We introduce a basic version of our localization algorithm and two extensions. One helps to improve calculating the position of relocated products while the other one fosters a faster localization using a smaller number of user data sets. We discuss the results of various simulation runs which give evidence that our system has a good potential to work in practice.**

***Index Terms*—Automated Localization, Real-Time Location Sensing, Intelligent Data Analysis, Mobile Services, Dynamic Leaky Accumulation, Softmax-based Inference.**

I. Introduction

In recent years, the rapid development of mobile technology has created unprecedented opportunities to realise intelligent environments. In particular, the wide presence of smartphones in our daily life with their powerful sensors and processing capabilities creates new opportunities to support the users with context-aware systems [1]. These types of applications use information gathered from the users' environments to support their decision making process in various fields, such as mobile recommendations. A *Mobile Recommender System* (MRS) analyzes information retrieved from the environment in which a user is moving through, and uses the analysis results to provide meaningful suggestions [2]. The recommendations can be made based on the information collected from the surrounding context and the user behavior.

A key element of context information for an MRS is the real-time position of the user, which is basically the current location of her/his smartphone. For context-aware outdoor systems, global navigation satellite systems like GPS provide this information, e.g., the pathfinding algorithm of Google Maps [3]. Until recently, the development of indoor MRS was hampered due to the lack of accurate smartphone-based indoor positioning technology. However, new indoor *Real-Time Locating Systems* (RTLS) are emerging that are able to detect the exact location of a user's smartphone with an accuracy of a few centimeters. For instance, the ultrasound-based Forkbeard technology [4] is a novel RTLS solution promising to locate a smartphone with a precision of less than 10 *cm*.

A typical MRS application using an indoor RTLS is navigation support in shops helping customers to find products as well as giving them useful recommendations. Similar to the navigation systems in vehicles, customers are provided with a path that guides them to the places where desired goods are stored. The realization of such a system, however, requires the accurate position of products, while in many shops this type of information is not available yet. The reason for that is that only two solutions seem feasible [5] that are both not optimal: One possibility is to attach special computer-readable tags to the products in a store and use them to let the RTLS track their exact positions. The provision of the products with such tags, however, is a major cost factor. The other solution is to register and update the position of the products manually. Yet, this needs a lot of human effort, in particular, if products are regularly relocated which is the case in many grocery stores [6]. Considering indoor localization, existing work is mostly focused on coarse-grained indoor navigation in stores, e.g., to guide the customer by means of different sensor types on the user's smartphone and basic localization technologies (e.g., signal strength measured at various places) [6]–[9].

In this paper, we aim at proposing a cost-efficient approach to automatically locate products in retail stores. Instead of tagging products or registering their positions manually, product localization is performed using two sets of customer data: the set of positions at which the customer stops while shopping, and the list of items purchased by her/him. Then, we accumulate these data pairs over a large number of customers. Since the customers can purchase products only by passing through the location of products, the accumulation reveals distinct correlations between items and places at which their buyers stop. These correlations are utilized to infer the positions of the goods. To realize this, we propose a

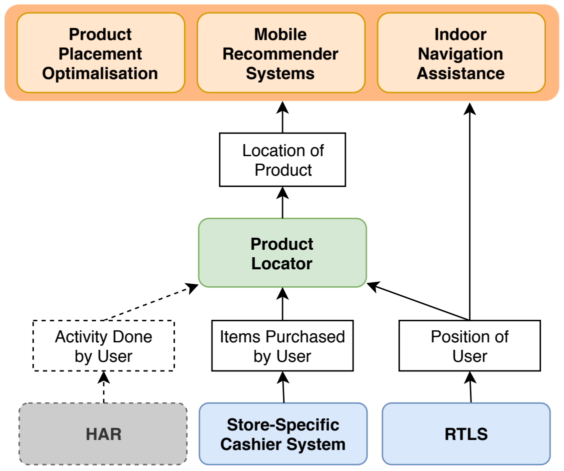

Product Placement Optimalisation
Mobile Recommender Systems
Indoor Navigation Assistance
Location of Product
Product Locator
Activity Done by User
Items Purchased by User
Position of User
HAR
Store-Specific Cashier System
RTLS

Fig. 1: The scope of Product Locator: from processing customer input data to potential applications

basic *Customer Score Accumulation* algorithm to accumulate customer data and to infer product locations based on the accumulation results. Further, we introduce two extensions of the algorithm (i.e., *Leaky Customer Score Accumulation* and *Softmax-based Inference*) to improve position calculation for relocated products, respectively to reduce the number of customers needed to infer the location of products.

Considering a significant number of errors in the customer data sets, e.g., stopping at places without picking up items, our simulation-based evaluation results show that 99.9% of the 8,000 products in a typical large Norwegian grocery store [10] can be correctly located after aggregating the data of around 12,000 customers. Assuming 1,000 customers a day, our algorithm needs to run about 12 days to calculate the location of items, which seems practically feasible.

The rest of this paper is organized as follows: In Sect. II, we give an overview of our aggregation and inference approach and discuss its scope. Thereafter, we present the approach in Sect. III followed by introducing both, the simulation technology used and the results revealed by the various test runs in Sect. IV. The article is completed with a discussion of related work in Sect. V followed by some concluding remarks in Sect. VI.

II. Overview and Scope

The scope of our approach is shown in Fig. 1. Intelligent location aware systems, such as mobile recommender systems, product placement optimization, and customer guidance, rely on the current position of users that can be obtained using an RTLS. As mentioned in the introduction, MRS also needs the location of the offered products which can be retrieved with our *Product Locator*. For a large number of users, the Product Locator collects data sets consisting of the places at which the customers stop while passing through the store as well as the list of items, they purchase. The customer stops are deduced from the RTLS data, while the list of purchased items can be obtained from the cashier systems or special apps provided by the stores. In addition, the activities done by the user during shopping can be a useful source of information for the Product Locator. In the rest of this section, we discuss these three sources of customer data and their relevance to our approach.

A. User Activity Recognition

Interpreting sensor inputs of mobile phones to find out certain activities is of growing importance, e.g., in health care [11]. This is usually done by *Human Activity Recognition* (HAR) systems that use pattern recognition algorithms to classify human activities from sensor data. Recently, the adoption of machine learning algorithms in HAR research have shown great results [12] such that HAR classifiers with an accuracy of greater than 90% can now be provided (see [13]–[16]).

The development of HAR systems, however, tends to be complex. In our context, it would be nice if one could detect whether a product is picked up or not from the movements sensed by the customer's smartphone. Yet, to allow an HAR system to learn the relation between sensor data and this activity, one needs example data sets for the activity "item pickup". Unfortunately, we could not find any openly available HAR data sets for this activity, and creating our own set would be a highly complex and laborious challenge. Therefore, we decided to use a much simpler way to recognize the picking up of items in a shop.

We simply record stops performed by customers while they are in the store. There are plenty of openly available datasets containing the activity "stop" and/or "standing" such that an HAR model trained on these datasets would provide the required stop locations. Using an RTLS, however, even a simpler approach might suffice: The stop activity is classified as "staying within a small area for a certain amount of time", e.g., staying in an area of one meter diameter for at least three seconds. Since customers perform those stops when they pick up products to buy, this classifier seems to be a good replacement for more complex activity recognitions.

Nevertheless, the price for the simplicity to just register stops instead of using more complex HAR mechanisms is a potentially larger number of errors in the customer data sets. For example, customers can also stop at places at which they only look at certain products but decide not to buy them, or simply conduct other activities like checking their phones. Moreover, they may pick up products without stopping such that the location of a product pickup is not registered by the RTLS. The simulations carried out, however, revealed that our algorithm is sufficiently robust against these errors. The only effect is that the number of customers needed to locate all products correctly is slightly growing but, the errors do not lead to false localizations (see Sect. IV-B).

B. User Position Detection

Most existing indoor RTLS are not sufficiently precise for applications that need an accuracy at the level of less than a meter. They are based on different radio communication technologies such as UWB, RFID, Bluetooth, Ultrasound, Visible Light, SigFox, and LoRA. Among these, Ultrasound-based systems seem to be particularly promising since they are less affected by interference from metallic objects than electromagnetic methods. They are less influenced by opaque objects in the environment than optical technology while being cheaper to realize [17]. An RTLS usually comprises of fixed units located at particular points in a closed room which receive wireless signals from tags or badges attached to persons or objects of interest. The units measure the arrival times of a signal which vary due to the different distances between the units and a tag transmitting the signal. From the travel time spread and the knowledge about the locations of the fixed units, the position of the signal transmitters can then be computed using triangulation.

A practical and efficient solution to locate a user in a confined area such as a retail store is to utilize the microphone of his/her smartphone as a localization tag. This is, for instance, done by the Forkbeard technology [4]. It uses fixed units which are usually installed at the ceiling of a room and emit ultrasound signals in precisely coordinated intervals. The various signals are received by smartphones in the environment which measure the time lags between the signals. Further, the fixed units send their exact positions in the room such that the smartphone can triangulate its own position. This technology promises to reach a precision level of at least 10 *cm*. For retail stores, this precision is more than enough since a location accuracy of around 30 *cm* is sufficient to differentiate between different product-containing compartments in the store.

The user stop detection mechanism is realized as follows: An app in a customer's smartphone measures constantly its position using the RTLS. If the phone rests for a certain time in an area as discussed in Sect. II-A, this is classified as a stop and the position is stored in a data base. In this way, all places at which the customer stops while shopping are retained. When the shopping activity is finished, the data set will be sent in anonymous form (to preserve the user's privacy) to the server running the Product Locator.

C. List of Purchased Items

The second data set to be used by the Product Locator is the list of all items bought by the customer in the shopping environment. A way to achieve this is to let the customer's smartphone retrieve the list from the cash register used for paying. An alternative is to use special customer apps. Lately, a trend among grocery stores has been to provide such apps giving additional digital experiences to their customers, e.g., systems providing bonuses, sales or discounts based on products purchased [18], [19]. These apps register which items each customer buys every time she/he visits the store. This information can be easily used to create the list of products purchased by a customer during a single store visit. Further, it is simple to combine this list with the set of stops also registered in the smartphone. From a judicial point of view, this solution is also helpful since customers who do not want their purchases being stored for inferring product locations, can simply switch of the tracking functionality or avoid to use this app at all.

III. Product Locator

Before describing the product localization approach in detail, we give a short introduction to the various designators used in this section. We define a particular indoor *Environment* of interest as E. The space covered by E is restricted by artificial boundaries in which a finite set of positions and items can be found. The set of all unique *Positions* in E is described by the set P while I refers to all unique *Items* available in E. The positions of the various items in E are described by the *Localization* function $L : [I \rightarrow P]$ that maps each item to its actual position. L is not an one-to-one mapping such that a position p can be assigned to several items.

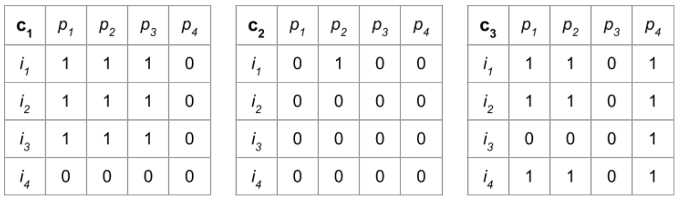
Our approach accumulates the data of a large number of customers $C = \{c_1, c_2, \dots\}$ moving in E over time. For a single customer $c \in C$, we define $P_c \subseteq P$ as the trajectory of positions, c passes while moving through E for shopping. The set $S_c \subseteq P_c$ is the set of positions at which the RTLS registers *Stops* for c. Further, we define $I_c \subseteq I$ as the set of items purchased by customer c during a traversal through E. Based on that, we can now define for a customer c the pair of *data sets* $ds_c \triangleq \langle S_c, I_c \rangle$ that will be utilized by the Product Locator.

In the following, we describe how the user data sets ds_c are analyzed and processed in order to obtain the localization mapping L for all items $i \in I$ in E. Our inference algorithm consists of three steps:

1) Calculate the score for each customer $c \in C$ from her/his data set $ds_c \triangleq \langle S_c, I_c \rangle$.
2) Accumulate the scores of all users in C and store the result in a data store.
3) Infer mapping $L : [I \rightarrow P]$ from the computation in step 2 by comparing the scores for each item that it is placed at a certain location, and return the position that has the highest score.

The three steps are described below. Thereafter, we introduce two extensions of the score accumulation step. The one is the *Leaky Score Accumulation* algorithm that weights newer customer data higher than older. This makes the algorithm more flexible against moving certain items to other positions in a store. The second improvement of the score accumulation is *Softmax-based Inference*. It uses the Softmax function [20] which makes the inference of correct item-to-position mappings faster and more reliable.

To clarify the various aspects of our Product Locator algorithm, we use a simple scenario. In a grocery store, bread is at position p_1, milk at p_2, eggs at p_3, and cheese at p_4. Further, we assume a user buying milk, bread, and eggs while a second one purchases only milk and a third one milk, bread, and cheese. For simplicity, we assume that all three customers stop at all the positions at which their purchased products are, and at no other positions. We can formalize that by using the sets $C = \{c_1, c_2, c_3\}$ referring to the three customers, $I = \{i_1\,(\text{milk}), i_2\,(\text{bread}), i_3\,(\text{eggs}), i_4\,(\text{cheese})\}$ describing the



c_1	p_1	p_2	p_3	p_4
i_1	1	1	1	0
i_2	1	1	1	0
i_3	1	1	1	0
i_4	0	0	0	0

c_2	p_1	p_2	p_3	p_4
i_1	0	1	0	0
i_2	0	0	0	0
i_3	0	0	0	0
i_4	0	0	0	0

c_3	p_1	p_2	p_3	p_4
i_1	1	1	0	1
i_2	1	1	0	1
i_3	0	0	0	1
i_4	1	1	0	1

Fig. 2: Matrices of the three customers c_1, c_2 and c_3 in the example.

four products, and $P = \{p_1, p_2, p_3, p_4\}$ representing the four positions of the products. The data sets for our three customers can then be defined as follows:

$$ds_{c_1} \triangleq \langle\{p_1, p_2, p_3\}, \{i_1, i_2, i_3\}\rangle, \quad ds_{c_2} \triangleq \langle\{p_2\}, \{i_1\}\rangle,$$
$$ds_{c_3} \triangleq \langle\{p_1, p_2, p_4\}, \{i_1, i_2, i_4\}\rangle$$

A. The Basic Algorithm

Let us assume that our indoor environment E comprises n different positions $P \triangleq \{p_1, \ldots, p_n\}$ at which products can be stored and offers m different items $I \triangleq \{i_1, \ldots, i_m\}$ for sale.

1) User Score Calculation: In the first step, we take the data gathered for a customer c and stored in form of the data set $ds_c \triangleq \langle S_c, I_c\rangle$. The set ds_c is transformed into a matrix M_c that contains a row for each item and a column for each position. Using m items and n positions, this matrix has then the size $m \times n$. In M_c, we mark all matrix elements considering an item purchased by c and a position at which c stopped as 1 and else as 0:

$$M_{c_{xy}} \triangleq \begin{cases} 1 & \text{if } i_x \in I_c \wedge p_y \in S_c \\ 0 & \text{else} \end{cases}$$

For the three customers in our example, this leads to the three matrices M_{c_1}, M_{c_2}, and M_{c_3} depicted in Fig. 2.

2) Accumulation of Customer Scores: In this step, the customer matrices M_{c_i} are accumulated to a *Data Store* matrix DS which can be achieved by simple matrix addition:

$$DS = \sum_{c \in C} M_c$$

For our example, the following matrix is computed:

$$DS = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \tag{1}$$

An advantage of the algorithm is that not all customer scores have to be received when running it. Instead, DS can be calculated based on just the currently available customer data. When data from a new customer c_{new} arrives, DS can be augmented by adding the corresponding customer score matrix $M_{c_{new}}$ to its previous version.

3) Inference of the Localization Mapping: The final step of the algorithm is inferring the locations of the items in the indoor environment from the data in matrix DS. To achieve that, we attach to each item the position that according to DS has the highest value.

Thus, we can define the localization mapping $L : [I \to P]$ as follows:

$$L[i_x \in I] \triangleq \text{choose } p \in P | \ \exists y \in \{1, \ldots, n\} : p = p_y \wedge$$
$$\forall l \in \{1, \ldots, n\} : DS_{xy} \geq DS_{xl}$$

If a product is placed at several positions in the store, we will locate only one of them which, however, is sufficient for indoor navigation and recommendation assistance.

Looking at our example data store DS (see eq. 1), we can only unambiguously infer that item i_1 is at position p_2 while the exact positions of the other items are not clear-cut after considering just three customers. The inferences would be getting unambiguous if some more customer results were considered. Nevertheless, we will see in Sect. III-C that there are improvements to the accumulation process possible that allow us to infer the correct positions even if only our three customers are considered.

B. Leaky Customer Score Accumulation

The goal of this optimization of the customer score accumulation is to reduce the *bias*, the system has for older data compared to newly collected customer scores. The data registered for a product in DS, i.e., the value of element e_{xy} for an item x at the position y, will become very large over time as more customers purchase the product, and to be able to do so, stop at its position. This can be seen as positive as long as the position of the product is never moved. If products, however, are regularly relocated which happens often in grocery stores [6], the algorithm needs quite long until inferring the correct position again. The reason is that many customer scores considering the new location of the item have to be accumulated until those considering the old place are outpaced. The *Leaky Customer Score Accumulation* algorithm allows us to mitigate this problem.

We define the so-called *leaky factor* α with $0 < \alpha < 1$. When we now add the score matrix M_c of a new customer c to the data store matrix DS, we reduce all the elements of DS referring to items bought by c and places that she/he did not visit. Using DS^{old} to describe the value of DS before adding M_c and DS^{new} for the one afterwards, we can express this operation as follows:

$$DS^{new}_{xy} \triangleq \begin{cases} DS^{old}_{xy} + M_{c_{xy}} & \text{if } M_{c_{xy}} \neq 0 \\ DS^{old}_{xy} \cdot \alpha & \text{if } M_{c_{xy}} = 0 \wedge \\ & \quad \exists l \in \{1, \ldots, n\} : M_{c_{xl}} \neq 0 \\ DS^{old}_{xy} & \text{else} \end{cases}$$

The intuition for this improvement is that for all items $i \in I_c$ purchased by c, the correct position for the items will most likely be among the positions $p \in S_c$ at which the customer stopped. Based on this assumption, all matrix elements in DS that describe products in I_c, and positions not in S_c, are less likely to refer to correct product locations. Thus, it is useful to reduce the values of these matrix elements. If an item is now moved to another place, there will be several customers buying it but not stopping at its old location anymore. Thus, the value for the old position will decline relatively quickly such that it will be faster passed by the value for the new location of the product.

Finding the right balance between prioritizing new over old data was done by empirical experimentation. The value $\alpha = 0.985$ achieved the best results in our tests.

The simulation of different values of α inspired us to a further improvement that we call the *Dynamic Leaky Score Accumulation*. Here we use a *confidence level* to indicate our trust in the data of a single customer c that is allocated her/his own leaky factor α_c. For instance, a customer buying one item and stopping at one location probably provides more accurate data than a customer stopping at 20 locations to buy a single product. Thus, the confidence level is based on two parameters, the number of items purchased, i.e., $|I_c|$, and the relation between the number of items purchased and number of stops performed by the customer, i.e., $\frac{|I_c|}{|S_c|}$. Our simulation runs showed the following dynamic leaky factor α_c to provide good results:

$$\alpha_c = \begin{cases} 0.750 & \text{if } 5 < |S_c| \leq 10 \wedge \frac{|I_c|}{|S_c|} \geq 0.5 \\ 0.650 & \text{if } |S_c| \leq 5 \wedge \frac{|I_c|}{|S_c|} \geq 0.5 \\ 0.985 & \text{else} \end{cases}$$

Thus, customers buying few products and stopping at most twice as often as the number of items purchased, are granted greater influence than other ones.

C. Softmax-based Inference

The intuition behind the second improvement strategy for the score accumulation in Sect. III-A2 is to amplify the differences between the values of the elements in our data store DS. Moreover, we want to consider the general numbers of stops at a certain position to evaluate the likelihood that a certain product resides there. Look for instance on the third row of the data store DS in our example (see eq. 1). There we have the value 1 in all of the first three columns such that we cannot claim a direct winner. On the other hand, the sums of the values in the first two columns of DS are much larger than the sum of the values in the third column. This indicates that fewer people not procuring item i_3 stopped at position p_3 than at p_1 or p_2 which makes it more likely that p_3 is, indeed, the place where i_3 is placed. The extension to the score accumulation algorithm presented here follows this consideration. It is based on the *Softmax* function (see, e.g., [20]) and will be carried out in three steps:

1) To amplify the more significant relations between items and positions, we transform matrix DS to DS^{exp} that uses exponential values. A problem to be solved is that the elements of DS may contain large numbers when the inputs of many users are stored such that computing the exponential value leads to an arithmetic overflow. To avoid that, we simply subtract the value ds_{max} of the largest matrix element in DS from all elements. Thus, no element will have a value larger than 0 and arithmetic overflow is prevented. The matrix DS^{exp} is computed as follows:

$$DS^{exp}_{xy} \triangleq e^{DS_{xy} - ds_{max}}$$

The use of the exponential value makes it possible to get rid of the zeros in data store DS since a product might be located in a position not yet visited.

2) Thereafter, we perform the last step of the Softmax function by normalizing the rows of the exponential matrix DS^{exp}. This is done in order to find, for each product, the probability distribution based on the stops performed by customers buying this product. The matrix reached in this step is named DS^{row} and computed as follows:

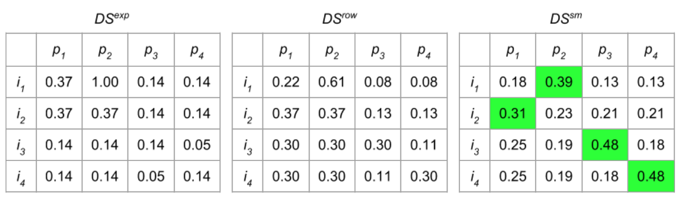
$$DS^{row}_{xy} \triangleq \frac{DS^{exp}_{xy}}{\sum_{l=1}^{n} DS^{exp}_{xl}}$$

3) Finally, we normalize the columns of matrix DS^{row} such that the sum of the values in each column is 1. The resulting *Data Store* DS^{sm} is computed as follows:

$$DS^{sm}_{xy} \triangleq \frac{DS^{row}_{xy}}{\sum_{k=1}^{m} DS^{row}_{ky}}$$

Thus, we realize the consideration discussed above, that a product is more likely at a position if relatively few people not buying it stopped there.

Then we take matrix DS^{sm} instead of DS in the inference step described in Sect. III-A3.



DS^{exp}

	p_1	p_2	p_3	p_4
i_1	0.37	1.00	0.14	0.14
i_2	0.37	0.37	0.14	0.14
i_3	0.14	0.14	0.14	0.05
i_4	0.14	0.14	0.05	0.14

DS^{row}

	p_1	p_2	p_3	p_4
i_1	0.22	0.61	0.08	0.08
i_2	0.37	0.37	0.13	0.13
i_3	0.30	0.30	0.30	0.11
i_4	0.30	0.30	0.11	0.30

DS^{sm}

	p_1	p_2	p_3	p_4
i_1	0.18	0.39	0.13	0.13
i_2	0.31	0.23	0.21	0.21
i_3	0.25	0.19	0.48	0.18
i_4	0.25	0.19	0.18	0.48

Fig. 3: The three matrices DS^{exp}, DS^{row}, and DS^{sm} of our example.

Starting with matrix DS of our example (see eq. 1), the extended algorithm provides the matrices shown in Fig. 3. In contrast to the result of the basic algorithm in eq. 1, the adapted algorithm makes it possible to infer the mapping L unambiguously since every row in matrix DS^{sm} has a unique maximum value (marked in green). Thus, the mapping L from items to positions can be correctly determined:

$$L[i_1] = p_2, \quad L[i_2] = p_1, \quad L[i_3] = p_3, \quad L[i_4] = p_4$$

For i_1, i_3, and i_4, this localization mapping can even be inferred if we demand a difference of 0.1 between the highest and second highest value to prevent wrong localizations.

IV. Evaluation

We evaluate the proposed approach through simulating the data sets. The issue is that real data traces for the discussed shopping use case are not currently available. For that, the necessary RTLS hardware for high precision indoor location sensing must be provided, and the existing store apps on the smartphones of the customers have to be extended to provide the list of purchased items per shopping. To evaluate the core functionalities of our proposed solution, and to learn more about it in order to fine-tune it (e.g., selecting the correct leaky factor α, see Sect. III-B), we carried out hundreds of simulations of various shop environments. To achieve that, we developed a suitable tool simulating indoor environments and the required customer data. For simulation, we considered data sets based on several distributions in order to cover many different realistic shopping scenarios. Therefore, we believe that the reported simulation results provide practical and relevant insights to our product localization solution. The first subsection describes the three modules of the simulation tool, in detail. Thereafter, we discuss the results of the evaluation runs.

A. Simulator

Our simulation tool consists of a creator for the shop environment, a creator for customer behavioral inputs, and the simulator core.

The *Environment Creator* module creates models of an indoor environment E to be simulated. For typical grocery stores in Norway, 8,000 items offered at 800 positions are realistic values to be used. The difference between the two numbers results from the fact that many products are pooled in shelves. From these inputs, the Environment Creator builds the sets I and P (see Sect. III) as well as the data store matrix variable DS (see Sect. III-A) with $|I|$ rows and $|P|$ columns.

The second module of the simulation tool is the *Customer Creator*. It is used to simulate customers moving through the simulated environment and purchasing the offered items. For each simulated customer c, the Customer Creator generates a random data set $ds_c = \langle S_c, I_c \rangle$ (see Sect. III). To simulate realistic customer behavior, the demand for certain goods differs vastly. This reflects that some products like low-fat milk are bought much more often than others, for example, mustard with truffles. We also like the number of items purchased by the customers to be in line with normal shopping behavior. To solve these two requirements without having access to real data of a retail store as a foundation for the simulations, the Customer Creator uses a truncated gaussian distribution when selecting the number of items purchased by a customer. This distribution is also used to select with which likelihood the customers buy a certain item.

As discussed above, we need to consider errors in the data sets since customers may stop at positions without buying the goods available there, and they might pick certain products on the fly without being detected as a stop by the RTLS. To consider these errors in our simulations, the Customer Creator works as follows: When creating a new customer c, it first determines the number of items the customer purchases using the corresponding truncated gaussian distribution. Thereafter, it selects which items the customer buys using the other gaussian distribution. This leads to the set I_c, followed by querying the Environment Creator for the correct locations of the selected items. Using these data directly as S_c would produce a "perfect" data set for the Product Locator, i.e., the customer would stop at exactly the positions at which the purchased products are located.

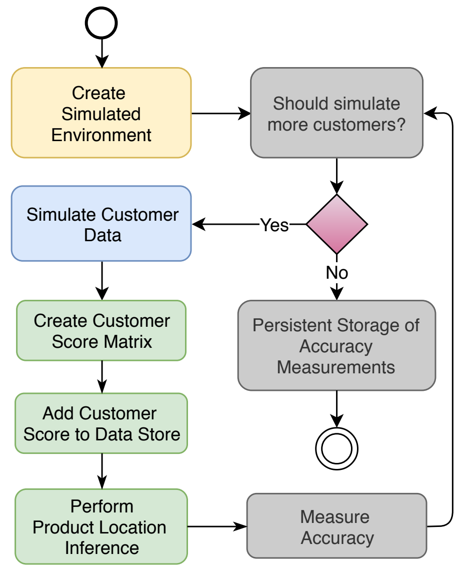

Create Simulated Environment
Should simulate more customers?
Simulate Customer Data
Yes
No
Create Customer Score Matrix
Persistent Storage of Accuracy Measurements
Add Customer Score to Data Store
Perform Product Location Inference
Measure Accuracy

Fig. 4: Flow chart illustrating the simulation tool

This perfect user set is then altered by an error function. At first, a random number of additional locations from the simulated environment is selected and added to set S_c. This function follows a certain distribution that can be parameterized prior to a simulation run. At second, to simulate the event that the RTLS misses stops at places at which items are picked, elements of S_c from the perfect data set are removed following also a parameterizable function.

The third module is the *Simulator Core* which is responsible for initiating both the Environment Creator and the Customer Creator. Further, this module realizes the Product Locator and feeds it with the data sets created by the Customer Creator. To give meaningful information about the accuracy of the Product Locator during the learning phase, i.e., its ability to predict the correct location for the products, the core module compares the produced function L with the real positions in predefined intervals and stores this information.

The different steps performed during a simulation run are depicted in Fig. 4. The simulation is started by the Environment Creator building an indoor environment. If more customers shall be simulated, the Customer Creator creates a new data set that may contain errors followed by the three steps of the Product Locator (see Sect. III-A). If the predefined interval is finished, moreover, the accuracy of the location mapping is checked and stored.

When all desired customers are simulated, the simulator

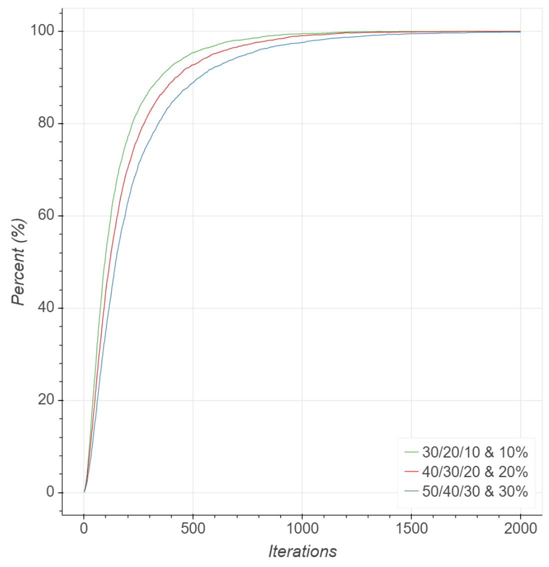


Fig. 5: The increase in accuracy as customer data is added to the system depending on different error rates.

terminates by storing the accuracy measurements in a JSON file. This data can be used to evaluate our proposed solution. We report about some simulation results in the following.

B. Results

Our Product Locator was thoroughly tested through hundreds of simulation runs, wherein various store configurations were used to further analyze the system behavior in varying environments. To evaluate the quality of our solution, we use the *accuracy* metric *acc*. Be $I_{loc} \subseteq I$ the set of all items that can be unambiguously assigned a location, since in the rows of the data store DS_{sm} (see Sect. III-C) the largest value is at least 0.1 larger than the second largest one. We then define the *accuracy* as $acc \triangleq \frac{|I_{loc}|}{|I|}$. The accuracy is measured and stored after adding the data of ten customers to the data store. Some of the simulation results are discussed in the following.

1) Testing the Basic Product Locator: First, we show the gradual increase in accuracy using what we claim to be "realistic" parameters. We simulated a shopping environment with 8,000 items, 800 positions, and 35,000 customers.

Besides stopping at the correct positions of the bought items, each customer can stop at various places without picking up any products. We use the so-called 30/20/10 likelihood pattern to select the number of such "erroneous" stops at which no products are picked up. It means that we choose a value between 0 and 10 erroneous stops with a probability of $\frac{4}{7}$. A number of stops in the interval between 11 and 20 of such stops is selected with a likelihood of $\frac{2}{7}$, i.e., half of the probability for the interval 0-10. With a likelihood of $\frac{1}{7}$, i.e., half of that for 11-20 stops, we select a value between 21 and 30 erroneous stops. When one of the three intervals has been chosen, the exact value within this interval is selected following a uniform distribution.

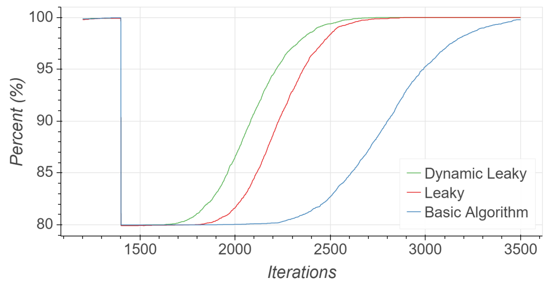


Fig. 6: The time to recover after moving 20% of the items in the environment.

Further, we assume with a probability of 10% that a stop at a correct position is not noted by the RTLS. Then, we get the hyperbola depicted in Fig. 5 with the *green* curve. With the simulation parameters described above, an accuracy of 99% is achieved after $8,420$ customers, 99.9% after $12,170$, and finally, all products were correctly located after $18,630$ customers.

2) Increased Error Rates: The second group of simulations intends to fathom the robustness of our algorithm by vastly increasing the likelihoods of additional stops and not detected pickings of products. We added the curves of these tests also to Fig. 5. The *red* curve describes a likelihood of 40/30/20 for additional stops and 20% for not registering pick ups, and the *blue* one is even more extreme assuming probabilities of 50/40/30 resp. 30%. It is interesting that the increased error rates do not change the hyperbola forms of the curves. The Product Locator still locates all items in the simulated environment, albeit with a slightly greater delay.

3) Handling Relocated Products: The third group of simulations considers the behavior of our algorithm when products are relocated. Figure 6 shows the results from simulations in which 20% of the items in the store were moved to a new location. All three simulations shown in the figure use the simulation configuration with the "realistic" parameters introduced in Sect. IV-B1. The *green* curve shows the results from the *Dynamic Leaky Accumulation* algorithm, described in Sect. III-B. The result using plain *Leaky Accumulation* is plotted in as the *red* curve, while the *blue* one depicts the results when using only the base algorithm without any leaky accumulation. These curves clearly show that the *Dynamic Leaky Accumulation* has a clear advantage against the simple *Leaky Accumulation* and even more against the simple algorithm when we have to expect significant relocation of items during the lifetime of the Product Locator.

4) Comparing the Score Accumulation Variants: Finally, we compare the customer score accumulation of the basic variant (see Sect. III-A2) with the *Dynamic Leaky Customer Score Accumulation* (see Sect. III-B), and the *Softmax-based Inference* (see Sect. III-C) for our "realistic" shop scenario without product relocations.

The *green* curve in Fig. 7 describes the behavior when using both Dynamic Leaky Customer Score Accumulation and Softmax-based Inference, while the *red* refers to customer

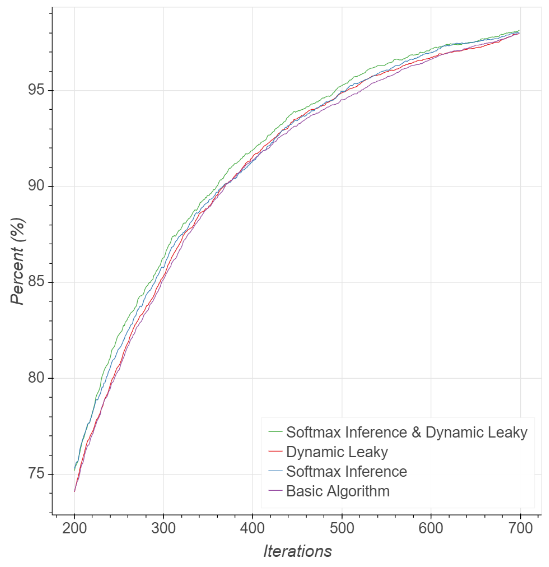


Fig. 7: Results of using different score accumulation variants.

score accumulation through Dynamic Leaky Customer Score Accumulation alone. Softmax-based Inference alone, is shown as a *blue* curve, and the basic algorithm is provided in *purple*. The curves reveal that the differences between the four scenarios are minimal. Just with relatively few customer data sets like in the example used in Sect. III, the Softmax-based extension gives a better accuracy score which becomes nearly irrecognizable when the data sets of more than 15,000 customers are considered. On the other hand, Softmax-based Inference has an advantage for localizing items that are rarely purchased. Thus, if a store has a fair number of such items, Softmax-based inference is more efficient than the basic algorithm.

V. Related Work

Being an important enabler for Mobile Recommender Systems (MRS), a fair amount of research has been conducted on indoor navigation systems. With respect to locating products in stores, most existing technologies propose manual tagging or registration of product locations performed by employees or system experts. To our best knowledge, previous work has neither been done on the automated localization of products in large stores nor on preserving the accurate location information when products are subject to relocation. Below, we discuss existing work of special interest.

Purohit et al. developed SugarTrail [6], a system providing the location of items and guidance to them in indoor environments without depending on existing maps. SugarTrail aggregates movement paths registered from users carrying mobile nodes utilizing magnetometers in addition to collecting data from stationary radios while traversing the indoor environment. The aggregated paths are used to construct Virtual Road Maps, which, in turn, can provide navigation assistance to items for customers in the store. The main contribution of SugarTrail is indoor navigation for stores, which is different from the goal of this paper, i.e., product localization.

Some other approaches leverage technologies such as computer vision, sensors and RSS to locate points of interests in indoor environments. Travi-Navi [7] combines high quality images and sensor readings from a *Guiders* smartphone and packs them into a navigation trace. This can be done, e.g., by a shop owner to provide navigation assistance to their stores. While moving through the indoor environment, the *followers* (customers) can follow the navigation trace defined by the Guider to the point of interest. In Canoe [8], the Received Signal Strength (RSS) is measured in various parts of a shop. Then, the observed RSS values are compared for directing users to points of interest. In Shopper Observer [9], the Redpin indoor localization framework which allows its users to create virtual fingerprints of locations, is used to find paths of customers in a shop which can be utilized, e.g., for product placement. In these approaches, the precision of localizations is lower. Therefore, they are better suited for scenarios where only relatively coarse-grain localization is needed.

In [21], Chia-Chen Chen et al. use RFID readers on smart devices together with RFID tags on products to recommend products to users according to their preferences, previous purchasing records, and current location. The recommendation mechanism works by a K-means algorithm, clustering customers based on their shopping behaviors. Augmented Reality (AR) is deployed in [22] to recommend healthy foods in grocery stores. This approach utilizes the camera on customer smartphones to both localize the customer within the store, and to display AR overlay *tagging* recommended products. The authors mention that product information is available in electronic product databases, however, not necessarily associated with their locations. This category of work is mainly focused on algorithms and techniques on intelligent product recommendation. In our view, the Product Locator proposed in this paper can be a good add-on to these approaches since it creates precise information about the position of products that can be leveraged to provide more intelligent user recommendations.

VI. Conclusion and Future Work

The paper presents an algorithm for product location detection in indoor environments. Our approach only requires the trajectory of stops and items purchased by customers while they move through an environment. The data collection can be done during the normal shopping routines of customers without any other participation than installing an app on their smartphones.

We developed a basic algorithm consisting of score calculation for individual customers, accumulation of the scores of various customers, and inference of product locations from the accumulated scores. Furthermore, we propose improvements to the score accumulation algorithm by *Static* and *Dynamic Leaky Accumulation* as well as *Softmax-based Inference*. Evaluating the various aspects of the algorithm, we found out that the *Dynamic Leaky Accumulation* can be very helpful in scenarios with relocated goods while *Softmax-based Inference* is more efficient when rarely purchased items have to be located and if only a relatively small number of customer data sets are available.

The next step is to create a simulator that considers the spatial aspects of a shopping environment. Using retail data sets, we plan to create a simulated shop layout and let customers "roam" through it simulating the trajectories performed while shopping. In this way, we can simulate varying customer behaviors, e.g., hasty purchases on the way home or cozy strolling being inspired by the available products. Also spatial aspects like the contorted ways performed when searching a hidden product can be simulated in this way. Thus, the extended simulator should allow us to test our approach with more realistic customer data.

Finally, when the Forkbeard technology is fully implemented, a prototype installation with a Norwegian grocery store operator is planned. This will make it possible to find out even more about user activity recognition, in particular, how long the waiting time of a user in an area should be in order to classify that as a stop. Moreover, we will investigate the accuracy of the assumptions we made in the simulations about the likelihood of errors in stop detection.

References

[1] Ö. Yürür, C. H. Liu, Z. Sheng, V. C. M. Leung, W. Moreno, and K. K. Leung, "Context-awareness for mobile sensing: A survey and future directions," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 68–93, 2016.

[2] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems: Introduction and Challenges*. Boston, MA, USA: Springer US, 2015, pp. 1–34.

[3] R. Gibson and S. Erle, *Google Maps Hacks*. O'Reilly Media, 2006.

[4] "Forkbeard Technology," https://www.sonitor.com/forkbeard/, accessed: 2018-12-04.

[5] P. Bolliger, "Redpin - Adaptive, Zero-configuration Indoor Localization Through User Collaboration," in *First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT)*. San Francisco, CA, USA: ACM, 2008, pp. 55–60.

[6] Purohit, A., Sun, Z., Pan, S., Zhang, P., "Sugartrail: Indoor Navigation in Retail Environments without Surveys and Maps," in *10th annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2013, pp. 300–308.

[12] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep Learning for Sensor-based Activity Recognition: A Survey," *CoRR*, vol. arXiv:1707.03502, p. 10 pages, 2017. [Online]. Available: https://arxiv.org/abs/1707.03502

[7] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-Navi: Self-Deployable Indoor Navigation System," *IEEE/ACM Transactions on Networking*, vol. 25, pp. 2655–2669, 2014.

[8] K. Dong, H. Ye, W. Wu, M. Yang, Z. Ling, and W. Yu, "Canoe: An Autonomous Infrastructure-Free Indoor Navigation System," *Sensors (Basel)*, vol. 17, no. 5, p. 996, 2017.

[9] M. Bourimi, G. Mau, S. Steinmann, D. Klein, S. Templin, D. Kesdogan, and H. Schramm-Klein, "A Privacy-Respecting Indoor Localization Approach for Identifying Shopper Paths by Using End-Users Mobile Devices," in *8th International Conference on Information Technology: New Generations*. Las Vegas, NV, USA: IEEE Computer, 2011, pp. 139–144.

[10] Norgesgruppen, "Innenfor flere varegrupper er matvareutvalget større i Norge enn i Sverige," https://www.norgesgruppen.no/presse/nyhetsarkiv, in Norwegian, accessed: 2019-01-27.

[11] M. N. K. Boulos, A. C. Brewer, C. Karimkhani, D. B. Buller, and R. P. Dellavalle, "Mobile Medical and Health Apps: State of the Art, Concerns, Regulatory Control and Certification," *Online Journal of Public Health Informatics*, vol. 5, p. 229, 2014.

[13] Z. He and L. Jin, "Activity Recognition from Acceleration Data based on Discrete Consine Transform and SVM," in *IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 5041–5044.

[14] Z.-Y. He and L.-W. Jin, "Activity recognition from acceleration data using ar model representation and svm," in *International Conference on Machine Learning and Cybernetics*, vol. 4, 2008, pp. 2245–2250.

[15] A. Khan, Y. Lee, and S. Lee, "Accelerometer's Position Free Human Activity Recognition using a Hierarchical Recognition Model," in *IEEE International Conference on e-Health Networking Applications and Services (Healthcom)*, 2010, pp. 296–301.

[16] O. Lara and M. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, pp. 1192–1209, 2013.

[17] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The Anatomy of a Context-Aware Application," *Wireless Networks*, vol. 8, pp. 187–197, 2002.

[18] Rema 1000, "Æ App," https://www.rema.no/ae/, in Norwegian, accessed: 2018-12-04.

[19] Trumf, "Trumf App," https://www.trumf.no/, in Norwegian, accessed: 2018-12-04.

[20] D. Heckerman and C. Meek, "Models and Selection Criteria for Regression and Classification," in *Thirteenth conference on Uncertainty in Artificial Intelligence (UAI)*. Providence, RI, USA: Morgan Kaufmann Publishers, 1997, pp. 223–228.

[21] C.-C. Chen, T.-C. Huang, J. J. Park, and N. Y. Yen, "Real-time Smartphone Sensing and Recommendations towards Context-awareness Shopping," *Multimedia Systems*, vol. 21, pp. 61–72, 2015.

[22] J. Ahn, J. Williamson, M. Gartrell, R. Han, Q. Lv, and S. Mishra, "Supporting Healthy Grocery Shopping via Mobile Augmented Reality," *ACM Trans Multimedia Comput Commun Appl*, vol. 12, p. 16:1–16:24, 2015.