Pervasive and Mobile Computing 46 (2018) 1–17

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Smartphone based approximate localization using user highlighted texts from images

Taeyu Im [a,*], Darius Coelho [a], Klaus Mueller [a], Pradipta De [b]

[a] Department of Computer Science, Stony Brook University, United States
[b] Department of Computer Sciences, Georgia Southern University, United States

ARTICLE INFO

Article history:
Received 22 December 2016
Received in revised form 24 December 2017
Accepted 12 February 2018
Available online 21 February 2018

Keywords:
Outdoor positioning
Content based image retrieval
Signal based positioning
Smartphone sensing
Database search
Pattern matching
Energy efficiency

ABSTRACT

In many application scenarios, an approximate location can suffice instead of achieving high accuracy of GPS, or other network infrastructure enabled localization. This can lead to design of localization systems low in resource consumption, and faster in obtaining a result. In this work, we design and implement a lightweight localization system, called WhereAmI, that can perform coarse localization with low resource requirement. The key intuition behind this work is that a collection of nearby textual signs in an image representing a user's surrounding forms a bag-of-words that provides a unique signature for her location. Due to the low performance of Optical Character Recognition (OCR) engine in outdoor settings, we develop a keyword-based positioning algorithm that can work even with partial errors in the detected texts representing business names. The partial errors in recognized business names are handled by using an n-gram-based text correction approach. We use a cloud based web service for offloading parts of the application workloads intelligently to save resources, like energy and network cost. The Android based prototype of WhereAmI is tested in uncontrolled environments. The experimental results show that WhereAmI can achieve 95% accuracy while consuming 20% less power than that of GPS. The proposed keyword-based positioning algorithm takes about 59 ms on average for returning the location.

1. Introduction

The proliferation of smartphones and wearables has allowed novel ways to determine user's location. Let us consider a scenario where a tourist is lost in a sprawling city with skyscrapers all around. The tourist has a smartphone, but she is running low on battery. GPS is not functioning well due to shadowing of the skyscrapers. Also, she is afraid of GPS draining the phone battery quickly. All she needs is an approximate idea of where she is on a map so that she can plan which direction to proceed towards her destination. Looking around she can see several business names which could easily act as a clue to give a sense of her location if she could ask someone familiar with the lay of the city. The collection of business names in an area often forms a unique signature of a place. Finding a feasible, resource efficient smartphone based solution to this scenario motivates the work in this paper.

Given the scenario described in the use-case, different approaches can be employed to find the location of the user without using GPS. The key intuition in all the techniques is to collect a signature marking the place, and then matching against geotagged signatures [1]. The signature in our example is hidden in the business names surrounding a user. The easiest way

* Corresponding author.
E-mail addresses: quocduy.vo@stonybrook.edu (T. Im), dcoelho@cs.stonybrook.edu (D. Coelho), mueller@cs.stonybrook.edu (K. Mueller), pde@georgiasouthern.edu (P. De).

https://doi.org/10.1016/j.pmcj.2018.02.004
1574-1192/

for the user to share this signature is by taking a picture of her surrounding using her smartphone camera. Then the picture content can be matched to find the location. The idea of localization using images captured by mobile devices have been explored under the theme called Mobile Visual Location Recognition [2–6].

Mobile Visual Location Recognition has progressed along two dimensions: one, matching image features to geotagged images; and second, finding texts in images serving as location context and finding geotags for the contexts. With advances in computer vision techniques, extracting and matching image features have become more efficient, but the computation cost, as well as, latency for computing a match is still high [7,8]. The alternative approach of extracting texts from images has gained in promise with the improvements in Optical Character Recognition (OCR) techniques [9–11]. However, depending on the complexity of the image from where the texts must be extracted, the accuracy of text recognition can drop significantly, thereby leading to low accuracy of locationing. Zamir et al. overcame the challenge of automatic business name recognition from complex image content by using both visual and non-visual (GPS, business directories) context related to an image [12]. On the other hand, Song et al. were able to detect texts from roadsigns with high accuracy since these are images with cleaner uniform background [11]. Vo et al. showed that even with partially correct texts that represent business names extracted from images, locationing can be performed as long as the number of textual signatures are reasonably high [10]. However, limitations of OCR capability pose a roadblock in pushing the limits of locationing accuracy in many of these text based techniques.

In this work, we address the challenge of accurate detection of textual signatures from an image by proposing a technique to predict correct texts from partially correct texts detected using OCR. The idea is to construct an n-gram dataset based on the names of businesses surrounding a location. Then the partially correct text can be matched with the n-grams to predict the correct business names. The main advantage of this approach is that it leads to faster location recognition as the n-grams can be computed offline, or cached. In addition, with more accurate textual signatures locationing accuracy improves significantly.

We have implemented the technique on top of the existing WhereAmI Android application. The WhereAmI application,[1] presented in Section 2, implements locationing by extracting multiple texts from an image using only OCR. Hence it suffered when texts have inaccuracies, leading to low locationing accuracy and slow response time due to longer matching time. In this work, we showed that applying the n-gram based approach for text detection on top of OCR, we can improve both accuracy and response time for locationing using texts from an image.

In summary, the main contributions of this work are:

- We present a technique for improving OCR-based recognition of texts, that represent business names, from an image.
- We present a complete design and implementation of an Android application that incorporates the improved technique for fast and accurate localization using multiple texts from an image.
- We conduct an evaluation with different datasets that show the effectiveness of the system in various outdoor settings in terms of localization accuracy, power consumption, and latency.

The remainder of this paper is organized as follows. Section 2 provides a background on the system design of WhereAmI, while introducing the details of the n-gram based text correction approach. It is followed by the implementation details in Section 3. Section 4 reports experimental evaluation of WhereAmI. We also discuss the pros and cons of WhereAmI in Section 5. Prior research related to this technique of localization are presented in Section 6. We conclude in Section 7.

2. WhereAmI system design

In this section, we present the design challenges of WhereAmI system. We also introduce the workflow and the detailed design of each component in the system.

2.1. Design challenges

2.1.1. Low performance of OCR on outdoor images

The key function of the front-end process is to extract textual signs from an image captured by a user using a smartphone. Off-the-shelf OCR engines work well for text extraction from scanned documents and for reading labels [13]. However, using OCR in natural images still remains a challenging problem. The OCR performance in such outdoor scenes degrades due to various uncontrolled parameters, such as variations in the font size and color, text alignment, uneven lighting, complex backgrounds, and moving objects [14].

2.1.2. Partial errors in text detection

Many OCR-based text recognition techniques for outdoor environment have been proposed with the aid of image processing techniques [15]. Despite these advancements, the text extraction process using OCR is error prone. Therefore, the proposed keyword-based localization algorithm must be designed to handle partial errors in keywords. The matching process must not depend only on exact match, but must use similarity scores which denote how closely an incorrect keyword matches the correct one in the database.

[1] WhereAmI App- http://tinyurl.com/nbd4aqe.

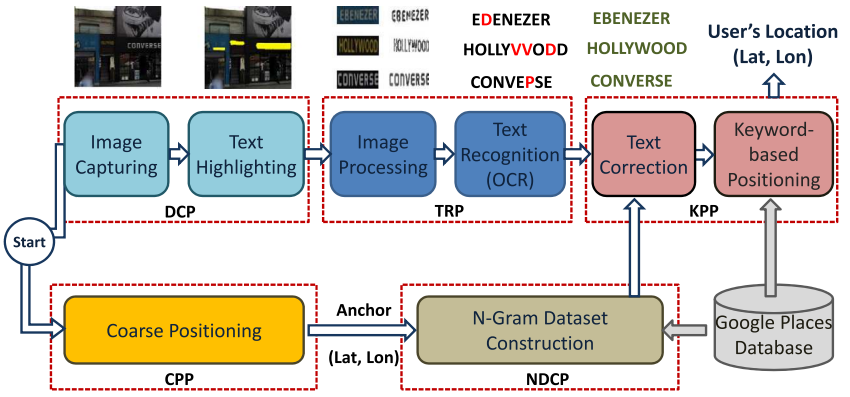


Fig. 1. WhereAmI system workflow showing the progression of steps in time.

2.1.3. Dataset construction using an anchor location

The success of the localization scheme hinges on the principle that a combination of names generates a unique signature of a place. However, if we consider popular store names around the world, then many of the names can occur in a group in the same location. For example, it is not unusual to find multiple coffee shops, like `Starbucks` and `Dunkin Donuts` in the same area. Although increasing the number of keywords in the combination can improve the accuracy, still a better option is to narrow down the search space with respect to the matching performance. The user's approximate location can be used as the anchor point to guide the matching process. This anchor point can be computed using coarse localization available on most mobile devices, such as via cell tower triangulation or the location of the current cell that the mobile device is connecting to.

2.1.4. Balancing the workloads between mobile devices and the cloud service

OCR libraries for mobile platforms are resource hungry and drain battery. The dataset construction process is also compute-intensive and time-consuming, compared to the keyword-based localization process. Executing the localization process while the data is still under-construction may fail due to the unavailability of the dataset, which degrades the success rate of the whole system. Therefore, performing these processes on mobile devices is not an efficient solution.

Recent mobile applications employ offloading computation to migrate certain workloads from mobile devices to cloud services. However, offloading computation can create problems in resource demands such as network bandwidth and workloads at the cloud service. In our system, therefore, balancing the workloads between mobile devices and the cloud service is necessary.

2.2. WhereAmI workflow

WhereAmI is designed as a fingerprint-based localization system, which consists of five main processes: Data Collection (DC), Text Recognition (TR), Coarse-grained Localization (CP), N-Gram Dataset Construction (NDC), and Keyword-based Localization (KP). The separation of these processes makes it easier to balance the workloads between the WhereAmI application and the cloud service. Fig. 1 shows the sequence of these processes in the workflow.

2.2.1. Data collection (DC) process

The workflow begins with a user starting the WhereAmI app on her Android smartphone. The app provides the user an interface to collect the textual signs by clicking a picture of her surrounding area or directly typing them into text fields shown in the interface. The users are guided to collect textual signs which consist of several properties making them ideal as location signatures, such as names of local businesses or buildings, prominent points of interests, and other fixed textual markers. While, spatially or temporally unstable signs, such as texts on vehicles or on advertisement banners, are easily identified and ignored.

Due to the low performance of OCR engine on a whole image, the users are required to highlight only the textual signs of interest. The highlights make it simpler to locate and construct a bounding box around the marked text area. We use flood fill algorithm to compute the extreme points of the highlight. Flood fill works by searching for a seed pixel(start point) for each highlight and testing whether its neighboring pixels belong to the highlight. The algorithm functions in a recursive manner and stops when it reaches the edges of the highlight.

In our method we store the point on the screen that the user touches at the start of every highlight and use them as the seed pixels of each highlight. This eliminates the seed pixel search step thereby optimizing the processing time of flood fill. We also handle overlapping highlights by merging and considering them as one continuous highlight. We use the computed extreme points to create a bounding box with additional 10 pixel padding and crop the box from the original image.

2.2.2. Text recognition (TR) process

Once the highlighted fragments are extracted, the text recognition process is triggered. Although the highlighted fragments are image segments containing keywords, there still remain some irrelevant contents, such as the background and other non-text objects. Removing these noises before performing the TR process on these fragments can improve the performance of the OCR engine. We first perform contrast enhancement to compensate for uneven lighting. Then we apply the Stroke Width Transform (SWT) to compute the text strokes while discarding the background objects [16]. By applying these techniques, the OCR engine can work with an acceptable accuracy.

2.2.3. Coarse-grained localization (CP) process

With respect to the matching performance in the localization process, the search space should be reduced. In our system, we use network location provider available on most mobile devices to acquire an approximate location of the mobile user. Although network location provider provides low accuracy (around 1620 m), but it works well for both indoors and outdoors, uses less power, and responds faster compared to GPS. Therefore, this approximate location can be used as an anchor to collect only businesses whose coordinates are located within the anchor's vicinity. The collected businesses are later used to construct a n-gram dataset. Since collecting business places and constructing the n-gram dataset are time-consuming processes, when the WhereAmI app is started, the coarse-grained localization process is also triggered and run simultaneously in the background.

2.2.4. N-Gram dataset construction (NDC) process

In spite of all the image processing, the text extraction process is not fool proof, and some keywords can have errors. In order to handle errors in the extracted keywords, we use an n-gram model to search for the best match of business names. The basic idea is to first construct a dataset of n-gram words from the given set of textual signs located in the vicinity.

Algorithm 1 *N − Gram Dataset Construction*

Require: A list of m place types provided by Google Places API or Daum Maps API $T = T_1, T_2, \ldots T_M$
Require: The anchor location: $Coord(A) = A_{lat}, A_{lon}$
Require: *SearchRadius*, *MinGram*=2

```
1:  for i = 1 to m do
2:     /*Search for places within the SearchRadius of the anchor area with type T_i */
3:     N_i = NearbySearchByType(T_i, Coord(A), SearchRadius)
4:     /*Put the neighbor places to NGramMap dataset*/
5:     PutToNGramMap(N_i)
6:  end for

7:  procedure PutToNGramMap(Set N)
8:  for each place p ∈ N do
9:     placeName = p.getPlaceName()
10:    for j = placeName.length to MinGram do
11:       Set grams = NGramBuilder(placeName, j)
12:       for each gram g ∈ grams do
13:          if nGramMap contains g then
14:             SET placeIDs = nGramMap.get(g)
15:             if placeIDs does not contain p.getPlaceID() then
16:                Set hashMap = nGramMap.get(g)
17:                hashMap.put(p.getPlaceID(), p)
18:             end if
19:          else
20:             Initialize a new HashMap ⟨String, Place⟩ hm
21:             hm.put(p.getPlaceID(),p)
22:             nGramMap.put(g, hm )
23:          end if
24:       end for
25:    end for
26: end for
27: end procedure

28: NearbySearchByType(type T, anchor A, radius R):
29: returns a set of places matching the specified type T within radius R distance from anchor A.
30: NGramBuilder(text, N): returns a contiguous sequence of N items from the given sequence of text.
```

Algorithm 1 presents the n-gram dataset construction (NDC) process. The process requires an anchor location $Coord(A)$ returned by the coarse-grained localization (CP) process, a *SearchRadius* parameter which is the maximum localization error of the CP process, and a list of place types T provided by Google Places API or Daum Maps API.

Although there is a nearby search query supported by the se APIs, the number of places returned in one API query is limited (60 for Google Places API, or 45 for Daum Maps API). In urban areas, therefore, many places may be missed in the returned list even they are located close to the anchor. Missing these places may degrade the performance of the localization process. In order to avoid missing these places, we execute each *NearbySearch* query separately according to place category. For each place type T_i in the list T, we retrieve a set of nearby places N_i which have the same type T_i (Line 3). Each set N_i then is used to create n-gram words and put into *nGramMap* which is a data structure that contains n-gram words and their corresponding places (Line 5).

Procedure *PutToNGramMap* shows how to construct the n-gram dataset, given a set of places. For each place name of place p in the list N, we build a set of n-gram words *grams*, where n is from 2 to the length of the place name (Line 11). For each gram word g of place p, if *nGramMap* already contains g, we add p into list of places which have the same gram word g. Otherwise, we create a new list of places corresponding to the gram word g, add the current place p to the list, and finally put the gram g and the new list into the *nGramMap*.

2.2.5. Keyword-based localization (KP) process

The localization process is divided into two phases: text prediction and keyword-based localization phase. The text prediction process takes as input a keyword obtained from the text recognition process and returns a list of business places, whose name matching the keyword. It requires the dataset of n-gram words and their corresponding business places *nGramMap*, which is constructed in the n-gram dataset construction process.

Algorithm 2 *Keyword − based Localization Algorithm*

Require: *NGramMap*, *NeighborRadius*, a list of n keywords $K = K_1, K_2, ...K_n$
1: **for** $i = 1$ to n **do**
2: P_i = *PlacePredict*(K_i) /* Predict K_i in *NGramMap**/
3: **end for**
4: $P = \pi_1^n(P_i)$ /* sort P_is in increasing number of elements*/
5: *ResultSet* = {} /*ResultSet contains sets of found places*/
6: **for** $i = 1$ to n **do**
7: *located* = *searchNeighbor*(P_i, i)
8: **if** *located* == TRUE **then**
9: break;
10: **end if**
11: **end for**
12: **procedure** Set⟨*Place*⟩ PlacePredict(String *keyword*)
13: **for** $j = keyword$.length() to *MinGram* **do**
14: *Set grams = NGramBuilder(keyword, j)*
15: **for** each gram $g \in grams$ **do**
16: **if** *NGramMap* does not contain g **then**
17: **continue** with the next gram
18: **else**
19: Map⟨*String* , *Place*⟩ $hash = nGramMap$.get(g)
20: List⟨*Place*⟩ $candidates = hash$.getValues()
21: **for** each place $p \in candidates$ **do**
22: *percent* = similarityPercent(g, p.getPlaceName())
23: p.setPercent(*percent*)
24: **end for**
25: **end if**
26: **end for**
27: **end for**
28: sort *candidates* in decreasing order of *percent* value
29: RETURN *candidates*
30: **end procedure**
31: **procedure** Boolean searchNeighbor(Set S, int *keyID*)
32: **for** $k = 1$ to $|S|$ **do**
33: $curResultSet = S[k]$
34: N_k = *GNeighbor*(Coord($S[k]$), *NeighborRadius*)
35: **for** $(z = 1$ to $n)$ && $(z \neq keyID)$ **do**
36: **if** $(N_k \bigcap P_z) \neq$ NULL **then**
37: $curResultSet = curResultSet \bigcup (N_k \bigcap P_z)$
38: **end if**
39: **end for**
40: **if** $|curResultSet| = n$ **then**
41: $ResultSet = curResultSet$;
42: RETURN TRUE;
43: **end if**
44: **if** $|curResultSet| > |ResultSet|$ **then**
45: $ResultSet = curResultSet$;
46: **end if**
47: **end for**
48: RETURN FALSE
49: **end Procedure**
50:
51: **GNeighbor(anchor** A**, radius** R**):** returns places within radius R distance from anchor A.
52: **similarityPercent(String** *gram*, **String** *placeName*)**:** returns the similarity percentage between two input strings

The text prediction algorithm, as shown in Algorithm 2 , begins by generating vectors of strings, P_i, for each keyword, K_i (Line 2). The *PlacePredict* function in Line 2 returns a set of business places whose name matching the keyword within *searchRadius* distance around the anchor point. It performs an approximate match between a set of n-gram words *grams* generated by each $keyword$ and the dataset of n-gram words *nGramMap* pre-constructed in the NDC process. The matching process is started with the longest gram g in the set *grams*. If the matching succeeds, the set of corresponding places whose names contain the gram g is obtained. Otherwise, the matching process continues with shorter grams until it can find the match.

Since there can be partially correct keywords, we approximately match the text where partially matching keywords or business names are returned along with a similarity score. As the number of the places predicted by the *PlacePredict* function increases, the algorithm will take longer to terminate, increasing system response time. Therefore, we reduce the size of the result set by computing the similarity percentage *percent* between the query keyword and its corresponding business names (Line 22). The similarity percentage is computed using distance which is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. Finally, we return a list of business places ordered according to their similarity percentage (Line 29).

In the localization phase, we combine these sets of business places to infer the user's location. Note that, the smaller cardinality of P_i indicates that K_i is a less common business name in the area. Therefore, beginning the search in the neighborhood of a least common K_i will converge faster. Hence we sort the P_is based on the cardinality of the P_is (Line 4).

In the procedure *searchNeighbor*, first a business name, say $S[k]$, from P_i is picked in the order they were retrieved. The *GNeighbor* function takes as input the location attribute of $S[k]$, and retrieves all the business names around $S[k]$ within *NeighborRadius* distance (Line 34). The *NeighborRadius* parameter is used to find the names which are clustered near each other and can lead to a solution.

If $S[k]$ is in the area where the user is, then the neighborhood business names vector, N_k, should contain the rest of the keywords. Line 37 searches for this match across the remaining sets of (P_z)s corresponding to the remaining keywords (K_z)s. If all the keywords are matched, the search terminates and the *ResultSet* contains all the businesses which the user marked and are in her surroundings (Line 42). We compute a bounding box using all the locations in the *ResultSet* and assign the midpoint as the user's location on the map. However, if $S[k]$ is not in the area where the user is, we try again with the next business place from the set S. We report failure to localize when there is no place predicted. In case of a failure, the user is sent a feedback to take another picture, and the process is repeated.

Although the complexity of the algorithm is $O(n^2\,|S|)$, we found empirically that typically 3 keywords are sufficient for the algorithm to locate a user's location.

3. WhereAmI system implementation

In this section, we present the implementation of WhereAmI system. We also discuss different working modes of the system with respect to power consumption, latency, accuracy and data network usage.

3.1. System implementation

We separate the system into two main components: the front-end is an Android app that runs on mobile devices (i.e. smartphones), and the backend component is designed as a web service that handles requests from the client app. The detail of these components are described as the following:

3.1.1. The front-end application

The front-end application is used to collect the input data for the localization process in the backend service. It is mainly responsible for coarse localization, image capturing, text area highlighting, image segment extraction, request sending, and location displaying. Depending on the working mode, the WhereAmI app executes different set of processes. The partly server-based working mode requires the app to recognize the keywords from the captured image and send only keyword texts to the backend service for localization. While, the server-based working mode directly sends image segments containing textual signs to the service. The standalone working mode allows the app to work independently without connecting to the server.

Since this implementation requires C++ source libraries, we use the Android Native Development Kit (NDK) to compile the libraries on mobile Android platform. In order to implement the highlighting feature, we use the OpenCV library which provides all of the image processing functionalities. The implementation of the image fragment cleansing uses the SWT source code from Kumar and Perrault [17]. Kumar's implementation additionally requires the Boost C++ libraries,[2] along with OpenCV. We use the Tesseract OCR engine,[3] which provides multi-language capability and has multi-platform support.

[2] Boost C++ libraries - http://www.boost.org/.

[3] Tesseract - https://code.google.com/p/tesseract-ocr/.

3.1.2. The backend cloud service

The backend service is mainly responsible for handling requests from the front-end application. There are three types of requests the service can handle: (i) dataset construction request, (ii) localization request with a set of image segments, (iii) and localization request with a set of keywords.

Since the dataset construction process is time-consuming, we construct the dataset right after starting the WhereAmI app. We implement a service running in the background of the WhereAmI app to manage the dataset construction. This service first obtains an approximate location as an anchor location using network location provider. This anchor location is later sent to the backend server to construct the dataset. When the backend service receives the dataset construction request, it starts constructing the dataset of business names.

In our implementation, we use both Google Places API and Daum Maps API to access a large database of business names along with their location coordinates. The reason of using both APIs is that Google Place database has a lack of business names, compared to the Daum Maps database, which is one of the most popular map service in Korea. Using the anchor location sent by the user, therefore, we first decide which API should be used to construct the dataset. If the anchor location is in Korea, we use the Daum Maps API to construct the dataset. Otherwise, we use the Google Places API.

Using these APIs, we implement the *NearbySearchByType* function in the dataset construction process (Algorithm 1) and *GNeighbor* function mentioned in the keyword-based localization process (Algorithm 2). The *NearbySearchByType* function allows to search for a specific type of business within a given radius from the anchor location. While, the *GNeighbor* allows to search for all business within a given radius for each specific business name.

Although there is a nearby search query supported by the se APIs, but the number of places returned in one API query is limited (60 for Google Places API, or 45 for Daum Map API). Due to this limitation, the *NeighborRadius* parameter in the *GNeighbor* function must be chosen carefully. If the radius is too small, then the business names extracted from the captured image may not be considered in the result. According to our preliminary experimental results, we use 20 m as *NeighborRadius* in our algorithm.

3.2. WhereAmI working modes

We allow the user to select which localization mode is suitable with respect to power consumption, latency, accuracy and data network usage. There are three different working modes one can select: Standalone WhereAmI, Server-based WhereAmI, and Partly Server-based WhereAmI. The detail of these working modes are described as the following:

3.2.1. Standalone WhereAmI

The standalone mode is used to allow the user to run WhereAmI application without a remote server. In this mode, all of the processes are executed on the mobile devices. Since the n-gram dataset construction process is time-consuming and compute-intensive, we run this process simultaneously as a background service in the standalone mode. However, each mobile device has to collect and store business places into its own database. No data is shared between the users, thereby the app has to construct the dataset when it reaches a new area.

3.2.2. Server-based WhereAmI

In the server-based mode, only the data collecting process and coarse-grained localization process are executed on the WhereAmI app. The text recognition process, n-gram dataset construction process, and keyword-based localization process are executed on the server in order to reduce the power consumption on mobile device and increase the response time. By constructing the dataset on the server, in addition, collected businesses can be shared among the users, which can reduce time required to construct the dataset in the later queries.

We upload the highlighted fragments obtained from the data collecting process at the app to the server. At the server side, the keywords in the fragments are extracted via text recognition process, and sent to the keyword-based localization process. In addition, since the n-gram dataset construction process is time-consuming, the anchor location need to be computed and sent to the server side as the WhereAmI app started.

3.2.3. Partly server-based WhereAmI

We also give the user an option to execute the text recognition process on the mobile app in cases where data usage is limited. The only difference between this mode and the server-based mode is that the text recognition process is executed on the WhereAmI app. Instead of sending segment images, which are highlighted fragments, the keywords are extracted by running the text recognition process on the fragments, and later sent to the server for localization. The bandwidth requirement in this mode therefore is reduced compared to uploading segment images as in the server-based mode. In addition, even a short message (SMS) will be adequate to upload the texts.

4. System evaluation

In this section, we evaluate WhereAmI with respect to localization accuracy, energy efficiency, latency, and data usage. We also show the ability to handle incorrect keywords of the localization techniques using n-gram-based text prediction.

Table 1
Experimental datasets with number of reference places (**places**) and number of pictures with *n*-Keywords (**n-K**).

Dataset	Places	1-K	2-Ks	3-Ks	Total
New York	52	10	119	42	171
Seoul	106	74	217	154	445

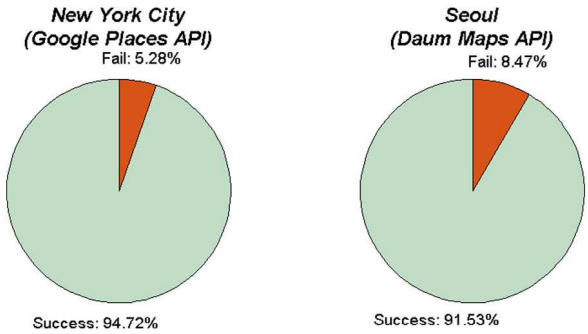


Fig. 2. The success rate of WhereAmI using correct keywords measured in New York City and Seoul.

4.1. Data collection

We conduct a real walk to evaluate WhereAmI with different experimental datasets collected by a Samsung Galaxy J5 phone in New York City and Seoul downtown. In total, we experiment WhereAmI at 158 reference places: 52 places in New York City and 106 places in Seoul downtown.

Since the number of store names affects the system performance, at each reference place we experiment WhereAmI multiple times using different number of store names. The maximum number of store names in each experimental image is limited to 3 due to the short distance from the smartphone camera to the set of store names.

At each reference place, we first use all of the store names from a captured image as an input set of keywords to compute the location and measure the localization performance. Next, we repeat this experiment by randomly removing only one store name from the set. We keep decreasing the number of the extracted store names one by one until the size of the set reaches to 1. In cases of evaluating the localization accuracy, the experiment is terminated as soon as the location is found. Table 1 summaries the experimental datasets we used to evaluate WhereAmI performance with respect to number of reference places and number of pictures with different keywords.

4.2. Localization accuracy

We experiment the localization accuracy of WhereAmI using both New York and Seoul datasets. We first measure the success rate of WhereAmI to show that the collection of geo-tagged store names can be used as a unique signature for localization. Next, we measure the localization accuracy to prove that WhereAmI can be used as an alternative outdoor localization method instead of GPS.

4.2.1. A collection of geo-tagged business names can be used for localization

Assuming business names are correctly recognized from the captured images, we measure the success rate of WhereAmI to show that a collection of store names appearing in one image can be used for localization. We also examine how the lack of business places registered in the databases reduces the success rate of the proposed localization technique. Finally, we show how the number of keywords affects the success rate of WhereAmI, given a condition that the input business names exist in the dataset.

We assume that OCR works perfectly so that the store names are correctly recognized from the images. We implement a text input interface in WhereAmI such that a user can input the names of the business places for localization. It enables controlled experiments with or without OCR errors.

Fig. 2 shows that WhereAmI can locate a user with an average success rate of about 93.1%, given a set of correct keywords (store names). Around 7% of the cases, WhereAmI fails to identify the user's location. This occurred due to the fact that some businesses used in the localization process are not registered in the database yet.

In addition, the success rate of WhereAmI experimented in New York City (94.7%) is higher than that observed in Seoul downtown (91.5%) on average. This occurred due to two main reasons. First, the change or removal of business places in the Daum Maps database in Korea happens more often compared to Google Places database in USA. Second, some business

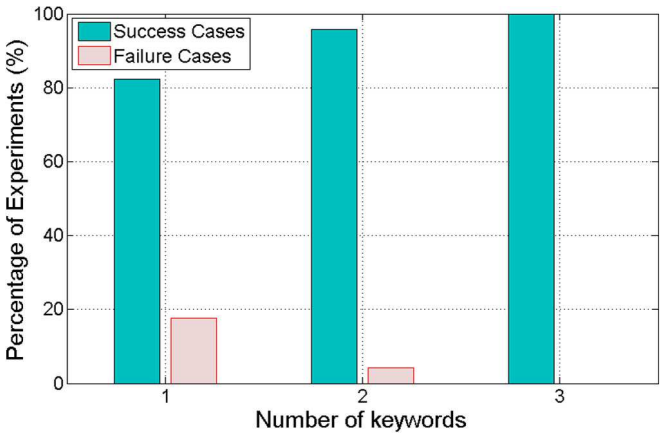


Fig. 3. The success rate of WhereAmI increases when the number of keywords increases.

names in Daum Maps database are different as compared to the signs. For instance, many Korean stores registered their business names in Korean, but their signs are written in English such as Starbucks, MC' Donald, etc.

Finally, we can see how the number of keywords affects the success rate of WhereAmI, given the condition that the input business names exist in the dataset. As shown in Fig. 3, the success rate of the matching algorithm improves as the number of keywords increases. When only one keyword is used, and it is not unique in the neighborhood, multiple locations are returned by the localization algorithm. It is not possible to identify which location is correct. The location, which may not be the correct location but closest to the anchor location, will be selected.

4.2.2. WhereAmI — An alternative localization approach

Next, we measure the localization accuracy of WhereAmI. We show that it can be used as an alternative outdoor localization method in cases where GPS module fails to obtain the location.

Toward this goal, we measure the location using GPS module simultaneously while localization the user using WhereAmI. The location errors of WhereAmI and GPS module are computed by measuring the distance from the locations returned by WhereAmI and GPS module to the ground truth location, which is manually pinpointed on Google Earth. Due to resolution limits of Google Earth, the ground truth location has an error of up to 0.5 m on average.

Fig. 4 shows a distribution of the localization errors of WhereAmI in terms of distances from the ground truth location. 88% cases WhereAmI can locate one's location, while localization fails in 12%. WhereAmI fails to locate the user due to the missing of the store names in the dataset (6.9% cases) and the use of unidentified keywords in the localization process (5.1% cases). When the store names extracted from the image becomes more erroneous, the text prediction process fails to identify the keywords. In 10% cases localization error was greater than 20 m. When only a single keyword is used, and it is not unique in the neighborhood, it leads to high localization error. Since multiple locations are returned by the keyword matching algorithm, it is not possible to identify which is the correct location. The one closest to the anchor location is selected, which may not be the correct location. Therefore, the users are required to provide more than one keyword to improve the localization accuracy.

Next, we show that WhereAmI can be alternatively used as an outdoor localization technique by comparing the localization error (LE) of WhereAmI to that of GPS using the following metric, *Relative Localization Error* (RLE).

$$RLE = \frac{LE_{WhereAmI}}{LE_{GPS}}$$

where $LE_{WhereAmI}$ and LE_{GPS} are the distances from the actual location of the user to the locations reported by WhereAmI and GPS respectively. The value of the ratio *RLE* indicates whether or not the location returned by WhereAmI is more accurate than GPS.

Fig. 5 shows the relative localization error between WhereAmI and GPS. When *RLE* is less than 1, the localization error of WhereAmI is smaller than that of GPS. The figure shows that in about 95% of the cases WhereAmI returns lower localization error compared to GPS. On average, WhereAmI achieves mean localization error of about 13.9 m, which is much smaller than that from the GPS (about 49.6 m). This is because the GPS module computes the location using the raw GPS signals received at the GPS module of the Android Framework. The received signals are affected by environmental conditions such as high density of buildings and bad weather condition. Whilst, localization using WhereAmI is based on getting the keywords and successfully matching them against the dataset.

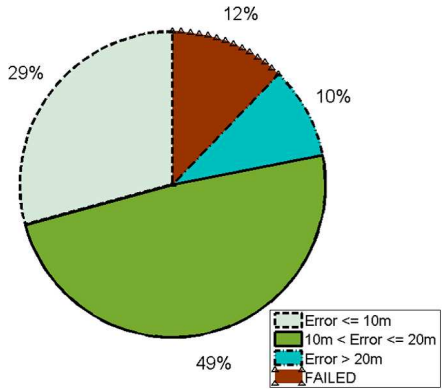


Fig. 4. Distribution of the localization error in terms of distance from actual location.

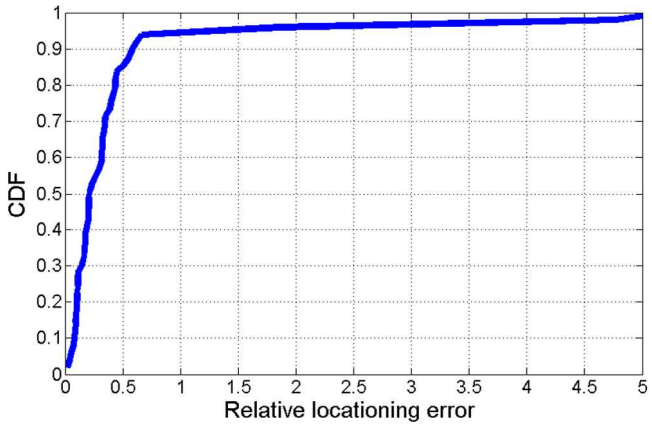


Fig. 5. Localization error of WhereAmI relative to localization error of GPS.

4.3. Power consumption

We measure and compare the power consumption of WhereAmI in three working modes: *Standalone WhereAmI*, *Server-based WhereAmI*, and *Partly Server-based WhereAmI*. We prove that the *Server-based WhereAmI* mode where the text recognition process, the n-gram dataset construction process, and the keyword-based localization process are executed on server, consumes less power compared to GPS and the other working modes. Finally, we evaluate the effect of the number of highlighted fragments on power consumption.

Using Seoul dataset, we measure the power consumption of WhereAmI using different number of highlighted fragments. We use PowerTutor to measure the power consumption of WhereAmI over 3G network [18]. The measurement is started from the time when the user starts the application until she receives her geo-coordinates information on her smartphone.

At each place, after measuring the power consumption obtained using WhereAmI, we profile the power consumption of GPS on the same test phone. After the GPS module is called, it starts receiving GPS signals to compute the location. Due to the environmental conditions, the GPS module waits for the signals until it receives enough information to calculate the location. Since the power consumption of the test phone fluctuates during the localization process and remains stable after the localization is successful, we can easily isolate the power consumption of the GPS module.

Fig. 6 shows the power consumption of WhereAmI in three different working modes, compared to GPS. On average GPS consumes 665 mW of power and WhereAmI consumes 673 mW of power in *Standalone* mode, 626 mW of power in *Partly Server-based* mode, and only 563 mW consumed in *Server-based* mode. The experimental results show that localization modes using server consumes 20% less power compared to that of GPS. In addition, Fig. 6 also shows that *Partly Server-based* localization consumes more power on the phone than *Server-based* localization. Although sending a group of texts consumes much less power than sending crops of fragments, the power consumption required for the text recognition process is much higher than that required to send the crops of fragments. Thus, offloading workloads to the server could save energy on mobile devices.

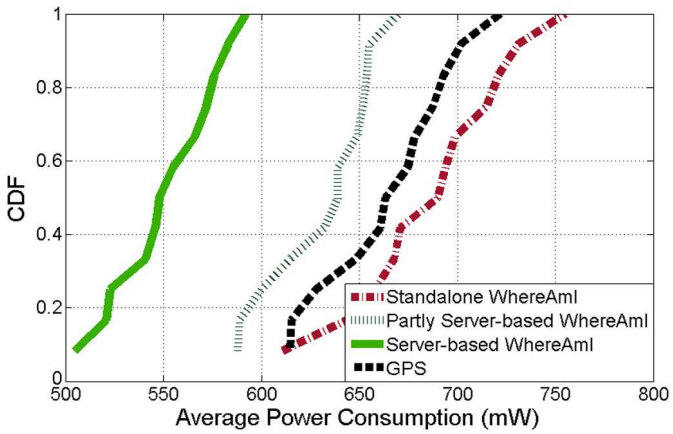


Fig. 6. Power consumption of WhereAmI in three working modes over 3G network, compared to GPS.

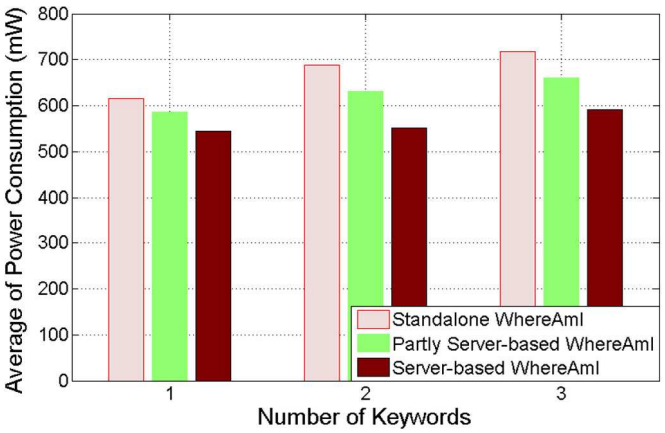


Fig. 7. Power consumption of WhereAmI in different working modes increases as the number of highlighted fragments increases.

Next, we evaluate the effect of increasing number of highlighted fragments on power consumption. Fig. 7 shows that when the number of highlighted fragments increases, the power consumption of WhereAmI increases in all working modes. This occurred due to the fact the text highlighting and fragment extraction are power-consuming process.

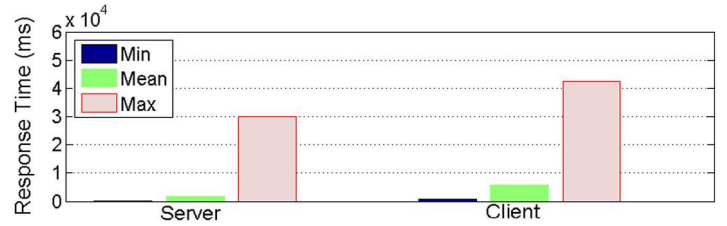
4.4. Response time

We evaluate the response time of WhereAmI with respect to the dataset construction process and the keyword-based localization process. We compare the response time of these processes executed on the client app and the server. We also compute the response time of the localization process with respect to the increasing number of keywords. Finally, the overall latency of the localization process using WhereAmI is compared to that of GPS.

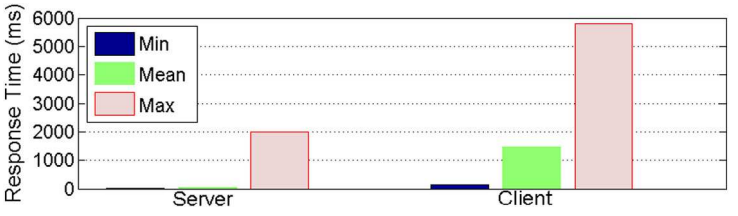
We record the processing time of these processes while measuring the localization accuracy using both New York and Seoul dataset. The processing time of each process is computed automatically and transparent to the user.

Fig. 8 shows the response time of the n-gram dataset construction process and the keyword-based localization process of WhereAmI on two working modes: on client and on server. As shown in the figure, execution of these processes on client consumes more time than on server. On average, it takes about 1.7 s to complete the dataset construction on the server side, while it requires about 6 s to complete the process on the client app. Similarly, the localization process on the server side is faster than that on the client side.

Fig. 9 shows the effect of the number of keywords on the latency of the localization process. As the number of keywords used increases, the response time of the localization process increases. In cases where none of the keywords is unique, the algorithm needs to search for more matches in multiple neighborhoods, hence taking longer. In the worst case, it could take close to 0.5 s to locate the user. On average, the localization algorithm takes about 59 ms for localization. The localization process can quickly predict the input keywords and identify the location of the user due to the n-gram dataset is constructed with a limit number of business names existing in the vicinity.



(a) N-gram dataset construction process.



(b) Keyword-based positioning process.

Fig. 8. The response time of the n-gram dataset construction process and the keyword-based localization process on client and server.

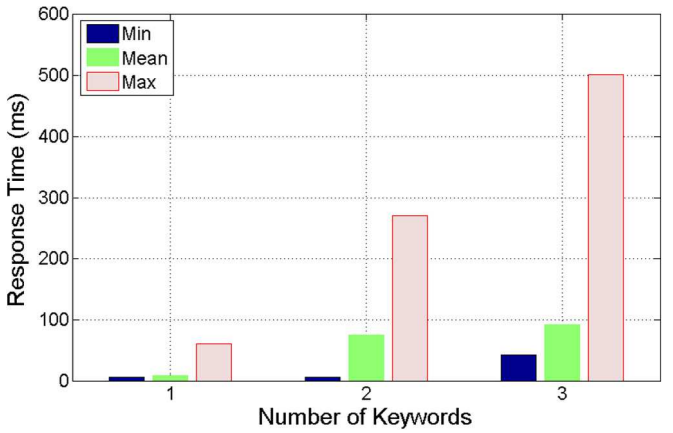


Fig. 9. The response time of the keyword-based localization process with respect to the increasing number of keywords.

We also measure how long GPS takes to identify the location. In comparison with WhereAmI, the GPS module needs around 12.5 s on average to complete the localization. GPS takes more time to localize due to the fact that it has to wait until it receives the satellite's signal before computing the location. In cases where the satellite's signal is blocked by an obstruction or interference, it need more time to complete the localization.

4.5. Cellular data usage

In this experiment, we evaluate how much network data the WhereAmI application needs to execute one localization request in two working modes: *Server-based* and *Partly Server-based WhereAmI*. We also measure the data bandwidth with respect to the increasing number of keywords.

Toward this goal, we use Seoul dataset, which consists of images at 800 × 480 resolution on the test phone (Samsung Galaxy J5). On average, each image has the size of 285 K and each image segment extracted from the capture image has the size of 11 K. Instead of sending the whole image to the server, we only send image segments for localization.

Fig. 10 shows the data usage of WhereAmI while increasing number of keywords in two working modes: *Server-based* and *Partly Server-based WhereAmI*. The figure shows that the *Server-based* mode needs more network data to identify one's location, compared to the *Partly Server-based* mode. The network data usage per one request consists of the input data uploaded to the server, the location returned from the server, and the data used to load the map with the user's location. On average, the *Partly Server-based* mode needs only 112 K per one request, while the *Server-based* mode needs 141 K per one request. In addition, when the number of keywords increases, the data usage in the *Server-based* mode increases but the

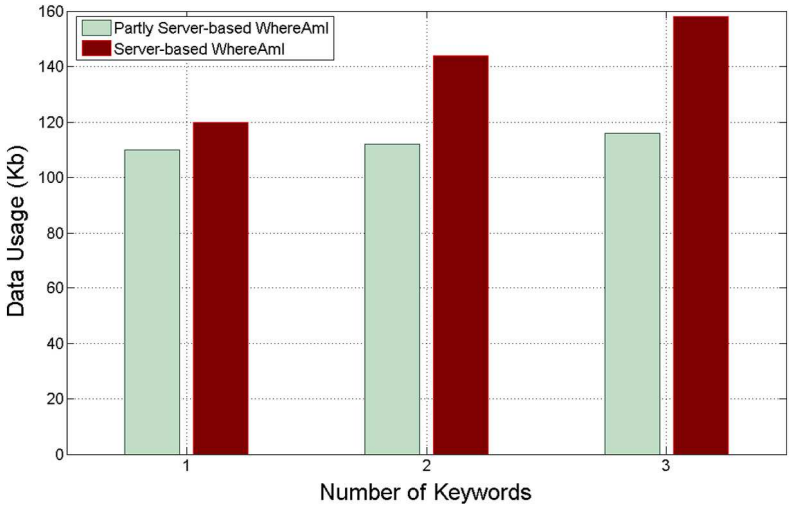


Fig. 10. Data usage of WhereAmI with respect to the increasing number of keywords in two working modes: *Server-based* and *Partly Server-based WhereAmI*.

data usage in the *Partly Server-based* mode does not increase much. This occurred due to the fact that transmitting a set of image segments requires more network data than transmitting an array of texts.

We conclude that there is a trade-off between the power consumption and network data usage in the WhereAmI app. The more network data is used, the less power consumed on the mobile device. We give the WhereAmI user an option to select which objective is most appropriate for the current status of her mobile device. If the phone battery is low, the user may select the *Server-based WhereAmI* mode to save battery life on the device, but more data usage is required. Conversely, when the user is running out of data usage, the *Partly Server-based WhereAmI* mode is selected to save the data usage, but more power is consumed on the mobile device.

4.6. Sensitivity analysis

In this experiment, we first present a user study to evaluate how fast users can identify the keywords in pictures. We also show the performance of the text detection using OCR. Finally, we measure the success rate of the text prediction and keyword-based localization process as the percentage of errors in input keywords increases.

4.6.1. User study on keyword identification accuracy

We performed a user study to validate the observation that users can quickly distinguish between permanent and temporary textual signs in a scene, and highlight the permanent signs. We first selected a set of 10 street view images from Google Street View and Daum Maps. Each image consisted of a mix of permanent and temporary textual signs showing as names of business. A group of 16 volunteers were individually presented with the images and instructed to highlight only the permanent textual signs in the image. The volunteers were given a time limit of 15 s per image to highlight the signs.

The results of our study showed that all the users marked the expected permanent business names in the images. There were only two users who marked temporary signs showing on a vehicle. Overall, the user study supports our human-in-the-loop approach for text detection.

4.6.2. Text detection accuracy using OCR

We evaluate the accuracy of Tesseract OCR library and show the effectiveness of the cleansing step (image processing) to improve text detection accuracy. We also generate an error model to represent the extent of error in the text recognition process with the cleansing process. This model later is used to introduce errors in keywords, which are used to evaluate how the text prediction process can handle error keywords.

We create our test set by using a mixture of images from the ICDAR datasets, Google street view, web images and images acquired using WhereAmI. Text fragments containing keywords are extracted from these images using WhereAmI's fragment extraction process. We create a dataset of text fragments which we use to evaluate our text recognition process. In order to measure the extent of errors introduced by the text detection process, we measure the number of characters that are recognized incorrectly, and report the percentage error. We conducted two experiments — OCR on image fragments with and without the cleansing step to determine the usefulness of the cleansing step to improve text detection accuracy using OCR.

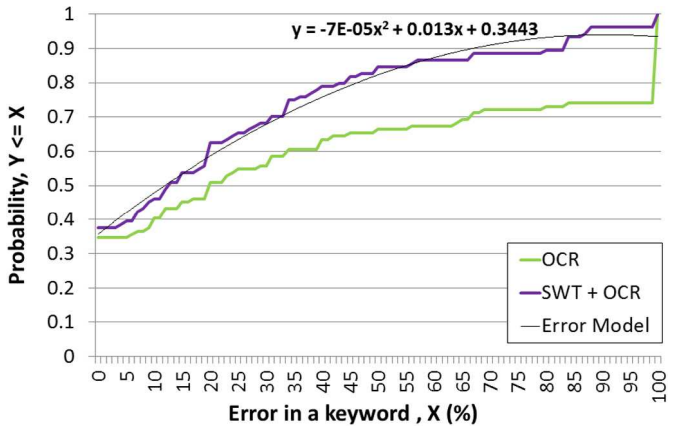


Fig. 11. Probability of the degree of error in a keyword while using OCR for text detection in two scenarios — with image cleansing before applying OCR, and without image cleansing. The degree of error with image cleansing and OCR is modeled, and the equation for error model is shown.

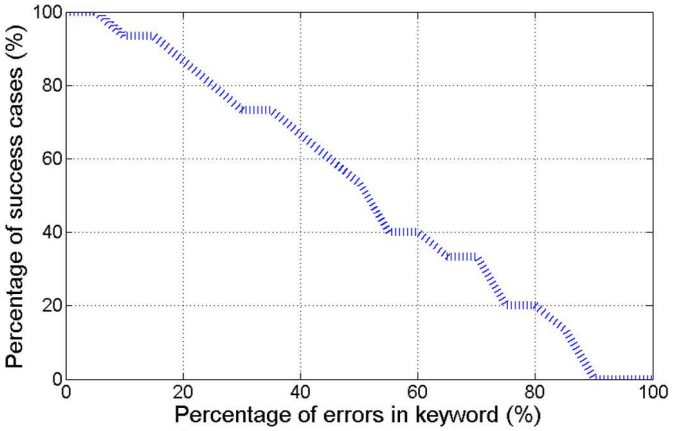


Fig. 12. The success rate of the text prediction process decreases as the keyword becomes more erroneous.

The results, as shown in Fig. 11, show the cumulative distribution of the percentage error in text detection for the two experiment scenarios. The figure shows that 37% of keywords in the dataset images could be recognized without any error, while 75% of the keywords could be detected with at most 34% error (percentage of mismatched characters in the keyword) using the cleansing step. But without the cleansing step, 34% of keywords could be recognized without any error and 75% of the keywords could be detected with at most 85% error.

We conclude that the accuracy of the text recognition is improved when the cleansing step is used. For further experiments, we also fit a curve to the CDF plot showing the error for the text recognition with the cleansing step (SWT + OCR). This represents the error model ($y = -7E\text{-}05x^2 + 0.013x + 0.3443$) for the OCR process used in our implementation.

4.6.3. Handling errors in keywords using the text prediction process

Although image cleansing process is used before the text recognition, there are still errors remaining in the recognized keywords. In this experiment, we verify the success rate of the text prediction process while increasing the percentage of errors in keyword.

We conduct this experiment using 15 test images from the Seoul dataset. Each image consists of at least three store names existing in the Seoul dataset. For each image, we first record a collection of the store names appearing in the image. Then we introduce different percentages of errors to each business name to generate incorrect keywords using the error model obtained in Section 4.6.2. Finally, we send each incorrect keyword to the keyword-based localization process, and record the result.

Fig. 12 shows how the success rate of the text prediction process is related to the percentage of errors in one keyword. As the percentage of errors in one keyword increases, the success rate of predicting the keyword decreases. The figure shows that a keyword with 50% of errors has the success rate of 50%.

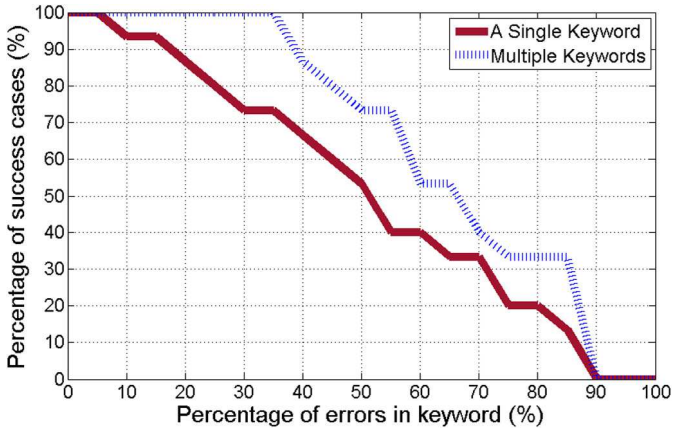


Fig. 13. The success rate of localization using multiple error keywords is higher than that of localization using single error keyword.

Table 2
Summary of the cases where WhereAmI performs more efficiently compared to other traditional approaches.

Techniques	Bad weather	Crowded areas	Requirements
GPS	Bad	Bad	No
WiFi	Independent	Independent	.WiFi Access Points .Wardriving process
WhereAmI	Independent	Independent	Business name(s)

4.6.4. Handling errors in keywords using the keyword-based localization process

In this experiment, we examine how the keyword-based localization process can handle incorrect keywords. We compare the success rate of the localization process in two scenarios while increasing of error percentage in keywords: localization with single incorrect keyword, and localization with multiple incorrect keywords.

Using the same set of images as in Section 4.6.3, we introduce the same percentage of errors (5%) to business names extracted from one image. We send the set of incorrect keywords to the keyword-based localization process and record the result. We continue the experiment using the same set of business names, but we increase the error percentage with a step of 5% and create new sets of incorrect keywords. We do the same experiment with the other sets of business names, and compute the average of the success rate of the localization algorithm according to the percentage errors.

Fig. 13 shows the comparison of the success rate of the localization using single incorrect keyword and localization using multiple incorrect keywords. When only one keyword is used in the query, the success rate of the algorithm is low due to several reasons. First, the input keyword is not unique, and therefore many place names contain the keyword may be returned. Second, the input keyword is not unique, also spelling errors drive the localization algorithm to irrelevant business names, which decreases the success rate. Although the text prediction process can improve the success rate in such cases, the keyword, which is corrupted badly after the recognition process, is not able to be predicted alone. Therefore, localization using single incorrect keyword returns in lower success rate, compared to localization using multiple incorrect keywords.

5. Discussion

In this section, we discuss the pros and cons of WhereAmI. We also summarize the cases where the system performs more efficiently compared to other traditional approaches.

5.1. Pros

Table 2 summarizes the cases where WhereAmI performs more efficiently compared to other positioning approaches. As shown in Table 2, WhereAmI can be used as an alternative localization technique in many cases where traditional approaches cannot work. For instance, one may be lost either in a crowded downtown area where GPS signals are blocked by obstacles (e.g. surrounding tall buildings, bad weather) or in a rural area where there is no WiFi access point. Only shop names or building names are visible to her. WhereAmI is able to figure out where she is by asking her to take a picture of surrounding textual signs. In addition, in unfamiliar areas where the user cannot read the language, the use of OCR engine in WhereAmI makes it easier to determine the textual signs for localization.

5.2. Cons

Despite its localization capability, WhereAmI has several disadvantages. First, the localization process of WhereAmI cannot perform automatically like other positioning techniques such as GPS or WiFi-based positioning. The system requires the users to interact with the app by capturing a photo of store names and highlighting the store names in the pictures for localization. It is inconvenient for the user and sometimes the users do not know which texts he or she should highlight for localization.

Second, the system does not return exactly the location of the user as other techniques do (e.g. GPS or WiFi-based techniques), but the locations of store names around the user found on the map. The user has to figure out himself where he is located on the map using the found store names.

Last, the system sometimes cannot find the store name due to two main reasons. The store may not have registered its name on the database yet. Also, the store name appearing in the captured picture may be different from the name stored in the database. For example, the store name shown in the picture is in Korean while it is stored in English and vice versa.

6. Related work

Research on fingerprint based locationing has shown use of different signature modalities for identifying a location. The signatures are often sensed using mobile device sensors, and then matched against a pre-constructed geo-tagged database of the signatures. In our work, we focus on the use of images, and specifically on the textual content of the images, as the signature. Hence in this section, we present our work in relation to three types of signature based locationing: (i) generic approach using data from different sensors, (ii) use of image features for locationing, and (iii) use of textual signatures in images.

6.1. Sensor data as signature

A comprehensive survey of different types of modalities for fingerprinting is presented in [1]. One of the widely explored modalities is wireless signal strength that has been applied for both indoor and outdoor localization [19]. A recent work showed that WiFi hotspots can be used for outdoor locationing [20]. Azizyan et al. showed in their work SurroundSense that logical labels of a place, represented by the ambient sound, light, color of a place, can be used as unique signature [21]. In UrbanEye, Verma et al. used a mixed set of physical features of the road, like speed-breakers, turns, etc. along with volatile landmarks, like GSM signal strength, cellular handoff characteristics, to represent the signature of a place [22]. Inertial sensors are used for localization in SmartLoc [23]. Map matching and dead reckoning algorithms are combined in APT for localization [24]. Dejavu uses different features of road landmarks (i.e. tunnels, bumps, bridges, and even potholes) as signatures to provide both accurate and energy-efficiency localization [25]. Other works have explored fingerprint-based techniques to improve the energy efficiency of GPS. For instance, RAPS proposes a rate adaptation of GPS sensing using hints gathered by inertial sensors [26]. CAPS uses the daily trajectory of users, and cell tower signals to improve over GSM based localization [27].

6.2. Image features as signature

One of the most intuitive ways of locationing is to match an image to a geotagged image to identify the location. This requires the user to post an image, which can be matched using different computer vision techniques to identify the location [2]. The idea was formally captured by the term Mobile Visual Location Recognition [3,6]. As computer vision techniques advanced, researchers showed improved techniques for extracting features from an image, like SIFT features, that can be easily compared to identify location [28,29]. Tian et al. showed that given a query street view image it is possible to use a reference database of geotagged bird's eye view images for geo-localization using convolutional neural networks (CNN) [8]. These approaches work on the entire image. They are typically computation intensive and expected to work offline. Hence these are not targeted for real-time localization with limited resources available on a mobile device.

6.3. Texts in image(s) as signature

Texts present in a picture often represent business names. If these business names are already geotagged by other business directories, like Google Places, then those provide a rich source for the signature. It can save time and resource on processing complex image features, thereby increasing the latency of recognition. The work in [30] extracts the words from in-store pictures and correlates these extracted words with their corresponding website to recognize the store its user is in. The system detects the name of the store using a repository of crowd-sourced pictures from different stores, each picture tagged with WiFi access points. Closest to our approach is the work in [9], which identify one's location by leveraging a human's perception of her surroundings. They also leverage surrounding textual signs such as street signs, local business logos, public transportation as cues for localization. Song et al. explored the possibility of a wearable positioning system, HoPS, where a user can take a picture of the road signs using Google Glass. The system extracts the texts from the image, collects cellular location information, and processes the text to identify the location [11]. Vo et al. also showed that a collection of business

names within an area also represents a signature of the area [10]. They could extract the texts representing business names with some user assistance and localize by matching against geotagged business directories. But inaccuracy in OCR text recognition limited the performance. This work extended the idea by better recognition of the texts from an image, and using it effectively for localization.

7. Conclusion

The ubiquitous presence of cameras in smartphones motivates the use of images captured by users for localization. However, visual location recognition techniques are compute-intensive. In this work, we present WhereAmI, a lightweight localization technique that uses texts appearing together in an outdoor image as a unique signature for localization. We propose a keyword matching algorithm that can handle partial errors in the detected text by matching approximately to texts from the database of store names. We have implemented WhereAmI on Android smartphones and evaluate its performance in terms of localization accuracy, power consumption, response time, and data usage. We also prove that WhereAmI can be an alternative outdoor localization system in cases where GPS fails to locate the smartphone users.

References

[1] Q.D. Vo, P. De, A survey of fingerprint-based outdoor localization, IEEE Commun. Surv. Tutor. 18 (1) (2016) 491–506.
[2] U. Steinhoff, D. Omerčević, R. Perko, B. Schiele, A. Leonardis, How computer vision can help in outdoor positioning, Ambient Intell. (2007) 124–141.
[3] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, E. Steinbach, Mobile visual location recognition, IEEE Signal Process. Mag. 28 (4) (2011) 77–89. http://dx.doi.org/10.1109/MSP.2011.940882.
[4] T. Guan, Y. Fan, L. Duan, J. Yu, On-device mobile visual location recognition by using panoramic images and compressed sensing based visual descriptors, PLoS One 9 (6) (2014) e98806.
[5] A.R. Zamir, A. Hakeem, L. Van Gool, M. Shah, R. Szeliski, Introduction to large-scale visual geo-localization, in: Large-Scale Visual Geo-Localization, Springer, 2016, pp. 1–18.
[6] W. Min, S. Jiang, S. Wang, R. Xu, Y. Cao, L. Herranz, Z. He, A survey on context-aware mobile visual recognition, Multimedia Syst. 23 (6) (2017) 647–665.
[7] J. Zhang, A. Hallquist, E. Liang, A. Zakhor, Location-based image retrieval for urban environments, in: Image Processing, ICIP, 2011 18th IEEE International Conference on, 2011, pp. 3677–3680. http://dx.doi.org/10.1109/ICIP.2011.6116517.
[8] Y. Tian, C. Chen, M. Shah, Cross-view image matching for geo-localization in urban environments, 2017. ArXiv preprint arXiv:1703.07815.
[9] B. Han, F. Qian, M.-R. Ra, Human assisted positioning using textual signs, in: Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, HotMobile '15, ACM, New York, NY, USA, 2015, pp. 87–92. http://dx.doi.org/10.1145/2699343.2699347.
[10] Q.D. Vo, D. Coelho, K. Mueller, P. De, Whereami: Energy efficient positioning using partial textual signatures, in: 2015 IEEE International Conference on Mobile Services, 2015, pp. 9–16.
[11] M. Song, Z. Ou, E. Castellanos, T. Ylipiha, T. Kamarainen, M. Siekkinen, A. Yla-Jaaski, P. Hui, Exploring vision-based techniques for outdoor positioning systems: A feasibility study, IEEE Trans. Mob. Comput. (2017).
[12] A.R. Zamir, A. Dehghan, M. Shah, Visual business recognition: a multimodal approach, in: ACM Multimedia, 2013, pp. 665–668.
[13] K. Jung, K.I. Kim, A.K. Jain, Text information extraction in images and video: a survey, Pattern Recognit. 37 (5) (2004) 977-997.
[14] H. Zhang, K. Zhao, Y.-Z. Song, J. Guo, Text extraction from natural scene image: A survey, Neurocomputing 122 (2013) 310–323.
[15] S. Milyaev, O. Barinova, T. Novikova, P. Kohli, V. Lempitsky, Image binarization for end-to-end text understanding in natural images, in: Document Analysis and Recognition, ICDAR, 2013 12th International Conference on, 2013, pp. 128–132. http://dx.doi.org/10.1109/ICDAR.2013.33.
[16] B. Epshtein, E. Ofek, Y. Wexler, Detecting text in natural scenes with stroke width transform, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2010.
[17] Text detection on Nokia N900 using stroke width transform, SWT. https://sites.google.com/site/roboticssaurav/strokewidthnokia.
[18] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R.P. Dick, Z.M. Mao, L. Yang, Accurate online power estimation and automatic battery behavior based power model generation for smartphones, in: Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES/ISSS '10, ACM, New York, NY, USA, 2010, pp. 105–114. http://dx.doi.org/10.1145/1878961.1878982.
[19] V. Honkavirta, T. Perälä, S. Ali-Löytty, R. Piché, A comparative survey of WLAN location fingerprinting methods, in: WPNC, 2009.
[20] J. Wang, N. Tan, J. Luo, S.J. Pan, WoLoc: WiFi-only outdoor localization using crowdsensed hotspot labels, in: Proc. IEEE INFOCOM, 2017.
[21] M. Azizyan, I. Constandache, R. Roy Choudhury, Surroundsense: Mobile phone localization via ambience fingerprinting, in: Proceedings of the 15th Annual International Conference on Mobile Computing and Networking, MobiCom, 2009.
[22] R. Verma, A. Shrivastava, B. Mitra, S. Saha, N. Ganguly, S. Nandi, S. Chakraborty, UrbanEye: An outdoor localization system for public transport, in: Computer Communications, IEEE INFOCOM 2016-the 35th Annual IEEE International Conference on, IEEE, 2016, pp. 1–9.
[23] C. Bo, X.-Y. Li, T. Jung, X. Mao, Y. Tao, L. Yao, SmartLoc: Push the limit of the inertial sensor based metropolitan localization using smartphone, in: Mobicom, 2013.
[24] X. Zhu, Q. Li, G. Chen, APT: Accurate outdoor pedestrian tracking with smartphones, in: INFOCOM, 2013.
[25] H. Aly, A. Basalamah, M. Youssef, Accurate and energy-efficient GPS-less outdoor localization, ACM Trans. Spat. Algorithms Syst. 3 (2) (2017) 4.
[26] J. Paek, J. Kim, R. Govindan, Energy-efficient rate-adaptive gps-based positioning for smartphones, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys '10, ACM, New York, NY, USA, 2010, pp. 299–314. http://dx.doi.org/10.1145/1814433.1814463.
[27] J. Paek, K.-H. Kim, J.P. Singh, R. Govindan, Energy-efficient positioning for smartphones using cell-ID sequence matching, in: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys, 2011.
[28] K.-H. Yap, T. Chen, Z. Li, K. Wu, A comparative study of mobile-based landmark recognition techniques, IEEE Intell. Syst. 25 (1) (2010) 48–57. http://dx.doi.org/10.1109/MIS.2010.12.
[29] N.N. Vo, J. Hays, Localizing and orienting street views using overhead imagery, in: European Conference on Computer Vision, Springer, 2016, pp. 494–509.
[30] R. Meng, S. Shen, R. Roy Choudhury, S. Nelakuditi, Matching physical sites with web sites for semantic localization, in: Proceedings of the 2nd Workshop on Workshop on Physical Analytics, WPA '15, ACM, New York, NY, USA, 2015, pp. 31–36. http://dx.doi.org/10.1145/2753497.2753501. http://doi.acm.org.proxy.library.stonybrook.edu/10.1145/2753497.2753501.