ABSTRACT
Location information is one of the key enablers to context-aware systems and applications for mobile devices. However, most existing location sensing techniques do not work or will be significantly slowed down without infrastructure support, which limits their applicability in several cases. In this paper, we propose a localization system that works for both indoor and outdoor environments in a completely offline manner. Our system leverages human users’ perception of nearby textual signs, without using GPS, Wi-Fi, cellular, and Internet. It enables several important use cases, such as offline localization on wearable devices. Based on real data collected from Google Street View and OpenStreetMap, we examine the feasibility of our approach. The preliminary result was encouraging. Our system was able to achieve higher than 90% accuracy with only 4 iterations even when the speech recognition accuracy is 70%, requiring very small storage space, and consuming 44% less instantaneous power compared to GPS.

1. INTRODUCTION
Location is one of the key context information and has enabled numerous context-aware systems and applications. To determine the location, today’s mobile devices use various sensors and related techniques such as GPS, Wi-Fi fingerprinting, cellular triangulation etc. All these approaches let the device figure out the location automatically and each of them complements one another by having distinct characteristics of availability, accuracy, infrastructure requirements, and resource consumption. However, most existing techniques rely upon the infrastructure support and/or heavy training for their proper use. For instance, on many smartphones, A-GPS (Assisted GPS) is typically used to get an initial location fix quickly. In this case, nearby cell towers or equipment directly connected to towers originate and deliver hints to smartphones so they can narrow down the search space. Without the help, it will take a long time to bootstrap the localization system. Some other localization techniques also require heavy training on the region of interest. Radio signal based fingerprinting techniques, such as ultrasound, Wi-Fi, and FM radio are representative examples.

In this paper, we propose an alternative positioning system by leveraging a human’s perception of her surroundings. The localization is computed based on what the user sees and tells the system.

One notable difference from prior art is that the acquisition of location information can be performed based on only local information. This is particularly useful for wearable devices. As depicted in Table 1, many such devices, including Google Glass and Apple Watch, do not have GPS, thereby the only way for them to obtain location information is to ask their paired devices, typically a smartphone. Moreover, the capability of completely local computation enables several unique use cases, which are articulated in 2.

Our localization methodology leverages textual signs we see every day. They include street signs, local business logos, public transportation signs, house numbers, decorative logos, etc. These textual signs offer several properties making them ideal as location signatures. First, they are prevalent as long as the area is reasonably populated [17]. Second, they are easily recognizable and readable with little ambiguity. Third, users can easily identify those spatially and temporally stable signs (Figure 1) and ignore those temporary signs that are not suitable for fingerprinting the location, such as text on a vehicle or on an advertisement banner. In contrast, automated approaches such as Optical Character Recognition (OCR) may have difficulties detecting many texts due to their diverse styles (examples in Figure 1), and OCR cannot tell which signs are stable. Another benefit of using text is the resultant signature database is small in size. As shown in Table 1, each entry corresponding to a sign takes only a few bytes. Therefore, the database of a large regional area can be stored locally. By integrating the database with offline maps, the entire positioning system can run locally without requiring GPS or Internet connectivity. The database can be constructed and updated by crowdsourcing without any domain knowledge or specialized equipment (e.g., by leveraging Google Street View, see Figure 4), making it much easier than collecting data for traditional maps [17]. For positioning, the user can, for example, read out several (e.g., 3 to 5) signs nearby she sees. The text is recog-

Table 1: Localization sensors on popular wearable devices.

<table>
<thead>
<tr>
<th>Device name</th>
<th>Release</th>
<th>Has GPS</th>
<th>Has Cellular</th>
<th>Has Wi-Fi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Glass</td>
<td>2013/03</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Samsung Gear 2</td>
<td>2014/04</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Apple Watch</td>
<td>2015</td>
<td>No</td>
<td>No</td>
<td>Partial**</td>
</tr>
</tbody>
</table>

* The sensor might exist but not yet accessible through the API.
** Can only communicate with its paired phone.
Table 2: Sign database for signs in Figure 1.

<table>
<thead>
<tr>
<th>ID</th>
<th>Text</th>
<th>Location</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Greene St.</td>
<td>40.729256, -73.995753</td>
<td>Road</td>
</tr>
<tr>
<td>2</td>
<td>West 4 St.</td>
<td>40.729256, -73.995753</td>
<td>Road</td>
</tr>
<tr>
<td>3</td>
<td>Rocco Restaurant</td>
<td>40.72938, -74.000157</td>
<td>Restaurant</td>
</tr>
<tr>
<td>4</td>
<td>23 St. Station</td>
<td>40.754528, 73.996623</td>
<td>Transportation</td>
</tr>
<tr>
<td>5</td>
<td>Downtown Brooklyn</td>
<td>40.754097, -73.982095</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Century 21 dept.</td>
<td>40.775921, -73.975867</td>
<td>Shopping</td>
</tr>
<tr>
<td>7</td>
<td>Regal</td>
<td>40.757065, -73.980707</td>
<td>Landmark</td>
</tr>
<tr>
<td>8</td>
<td>Century 21</td>
<td>40.774135, -73.950709</td>
<td>Cinema</td>
</tr>
<tr>
<td>9</td>
<td>iPhone</td>
<td>40.745528, -73.998623</td>
<td>House number</td>
</tr>
</tbody>
</table>

We motivate our approach by discussing several use cases.

**“Offline” Localization.** Bob’s grandmother is visiting him. Realizing she was lost in the unfamiliar city, she called her grandson. Using our system, Bob is able to quickly figure out where she is by asking her to describe her surroundings. In some cases, the same goal can be achieved by checking the street signs. However, street signs are not always nearby and not always available (e.g., in a theme park or shopping mall).

**Facilitating Location Search and Recollection.** Alice barely remembers the place where she met her friend last week. It was near a Starbucks and a Thai restaurant. Our system provides a way to effectively search this location based on the limited information.

**Offline Localization on Wearable Devices.** Many wearable devices such as Google Glass and Apple Watch do not have GPS (Table 1). Consequently, the only way for them to obtain location information is to ask their paired smartphones via Bluetooth. Therefore, if the user does not bring her smartphone, or if the phone runs out of battery, the wearable device cannot know its location. Our proposal fills this gap by requiring neither location sensor (GPS, Wi-Fi, cellular, etc.) nor Internet connectivity. It works in both outdoor and indoor environments.

**Infrastructure-free Indoor Localization.** Localization in indoor environments usually requires hardware beacons and/or heavy training. In our approach, location can be determined by leveraging existing textual signs such as room names and numbers. The sign database can be easily constructed by walking in the building and marking the locations of textual signs on the floor map.

### 3. LOCALIZATION SYSTEM DESIGN

Our proposed system consists of three components: the sign database, the user input module, and the localization algorithm. When a user enters a nearby textual sign, it is matched against the sign database to get a set of candidate locations. These locations are then fed into the localization algorithm to refine the user’s location. The above steps are repeated until the user’s actual location is pinpointed. We detail the three components below.

**Challenges.** At first glance, the proposed scheme looks straightforward. One key challenge, however, is to handle the inaccuracy that may appear in each step. First, the user may misread some signs. Second, the user may only enter part of the sign based on common conventions. For example, “King Harbor Seafood Restaurant” can be partly entered as “King Harbor” or “King Harbor Seafood”. Third, the user input module (e.g., the speech recognition engine) may make mistakes by translating the text to something different. Fourth, the same sign (e.g., “Bank of America”) may appear in multiple places thus causing ambiguities. Last but not least, a sign recognized by the user may not be in the database. We need robust algorithms to tolerate such inaccuracies. Another challenge stems from the fact that our system runs completely locally on mobile devices with limited storage, computation capability, and battery life. Therefore, it needs to be as lightweight as possible to minimize the resource consumption, but without compromising the accuracy and user experience.

#### 3.1 The Sign Database

The sign database contains precise locations of the textual signs. It can optionally contain tags for other value-added services. Based on the original database, our system generates an auxiliary data structure called token table by splitting each textual sign into separate words (i.e., tokens) and constructing mappings from each token to the set of signs in which the token appears. For example, from Table 2 we have “rocco”→{3}, and “regal”→{8} where “rocco” and “regal” are two tokens, and the numbers are sign IDs. The reason for introducing tokens is to tolerate errors. As will be described in §3.2, a user input is processed at the basis of each token instead of each sign as a whole. Therefore, if the user only enters part of a sign, or if there are mistakes in some words of the sign, the sign may still be identified because some tokens partially match.

#### 3.2 User Input Interface

Our system can accommodate any input interface as long as the method let users conveniently enter the tokens. For instance, speech recognition can be used. Or if an on-screen keyboard is available, one can simply type. We mainly focus on speech recognition (SR) as our primary input interface because microphone is one of the most prevalent sensor on mobile devices and SR is particularly suitable for wearable devices. When the user reads out a nearby sign, the SR engine translates her speech back into the text, which is then processed by the localization algorithm. SR performs a series of computations to find a match between a raw voice signal with words (or sentences) in the database. Typical SR processing chain includes three models: an acoustic model, a phonetic dictionary, and a language model. The acoustic model translates continuous raw voice signal to phones. The phonetic dictionary enables to map the phones to words. The language model decides legitimate sentences (e.g., word A cannot appear after word B).

In our initial design, we keep our language model simple in order to make the size of local database for SR as small as possible. As a result, the user input is recognized at the basis of each word token. Our SR is thus unaware of the structure of the text on signs as it only

---

11words” and “tokens” are interchangeable in this paper.
translates user’s speech into individual tokens. Here is the high-level description of token recognition procedure. Let the input be a sequence of tokens \{t_i\}. Their matched signs \{s_i\} can be quickly identified by looking up each \(t_i\) in the token table \((4.1)\). A token can match zero, one, or more signs in the token table. For each token \(t_i\), we define its frequency \(f(t_i)\) as the number of its unique matches \(i.e., \) signs containing \(t_i\) divided by the total number of unique signs in the database. Tokens with high \(f(t_i)\) are ignored based on a predefined threshold \(e.g., 0.001\) based on Figure 2, because they are unlikely to have enough discrimination power.

Also, note that the phonetic dictionary and language model can be changed dynamically in our case. Specifically, in the incremental update phase \((3.3)\), it only contains a small subset of all tokens in the database depending on where the user is. These operations can be done locally on most mobile devices, given the fact that nowadays it is common to observe multi-core CPUs on mobile platform and we have a relatively small number of tokens \(e.g.,\) several thousand per city \((5.3)\). All those factors discussed so far \(i.e.,\) simple language model and dynamically generated model files, contribute to reduce the complexity of the SR engine, thereby making it feasible to run on today’s mobile devices with reasonable accuracy, as to be demonstrated in \((5.3)\).

We admit that, as a completely offline approach, our SR accuracy can be potentially inferior to those cloud-based SR systems backed by large database. Also, tokens with same or similar pronunciations make it inherently difficult to distinguish them. To mitigate this issue, we also includes a feature allowing a user to correct the SR result at a per-token basis. The system displays its best guess, as well as \(k\) other candidate tokens and a “not listed” option so that the user can correct the result within a time window by using either voice command or physical buttons.

3.3 The Localization Algorithm

The localization algorithm works in a continuous manner. It takes as input a stream of signs \(S = \{s_i\}\). The output is either the estimated location or unknown indicating the location cannot be accurately determined due to a lack of input.

For each \(s_i\), we find its entries \(p_i,j (1 \leq j \leq k_i)\) in the database where \(k_i\) is the number of entries of \(s_i\) in the database. Assume we have \(m\) signs \(s_1,...,s_m\) from the user. Some signs might be incorrect. Also some may correspond to multiple locations \(i.e., k_i > 1\). In order to automatically filter out incorrect signs and locations, we leverage a key assumption that the majority of signs should have (at least) one entry appearing near the user’s true location because the user can see them. That is to say, for at least \(m_0\) signs, \(3j'\) such that \(p_i,j'\) is close to the true location, assuming we get \(m_0 \leq m\) signs correct. Consider an example in Figure 2. The user enters four signs \(s_1, s_2, s_4,\) and something else that is mistakenly interpreted into \(s_3\) \(e.g.,\) due to inaccurate speech recognition). \(s_1, s_2,\) and \(s_3\) appear at multiple locations in the database. Despite of these noises, \(p_{1,3}, p_{2,2},\) and \(p_{4,1}\) form the largest cluster, which with high probability corresponds to the user’s real location.

Following the intuition from Figure 2, we design a localization algorithm, with the high-level idea of searching for the largest cluster of signs by gradually increasing the size of the set of correct sign locations. As shown in Algorithm 1, For any subset of 2 possible locations, we measure their distance and count the number of subsets with distance smaller than \(D\), a pre-defined threshold. If we do not find any such subset, the location is unknown. If such subset is unique, the estimated location is the centroid of this subset of locations. If there are multiple such subsets, we repeat the above procedure for any subset of 3 possible locations, 4 possible locations,... until the subset size reaches a threshold \(T \leq |S|\). We empirically found this algorithm is more robust than off-the-shelf clustering algorithms such as k-means for our problem. We evaluate its accuracy in \((5.2)\).

**Algorithm 1 Localization Algorithm**

1: \(S = \{s_i\}, t = 2, \text{stop} = \text{FALSE};\)
2: while (\(\text{stop} \iff \text{TRUE} \&\& t \leq T\)) do
3: \(\text{found} = 0;\)
4: \(\text{for} (\forall S' \subseteq S, |S'| = t) \text{do}\)
5: \(r = \text{centroid of } S';\)
6: \(\text{if} (\forall s_i, s_j \in S', \text{distance}(s_i, s_j) < D) \text{ then}\)
7: \(\text{found}++;\)
8: \(\text{end if}\)
9: \(\text{end for}\)
10: \(t++;\)
11: \(\text{if} (\text{found} = 0) \text{ then}\)
12: \(\text{stop} = \text{TRUE}; \text{Output: unknown location};\)
13: \(\text{else if} (\text{found} = 1) \text{ then}\)
14: \(\text{stop} = \text{TRUE}; \text{Output: location is } r;\)
15: \(\text{end if}\)
16: \(\text{end while}\)
17: \(\text{if} (\text{found} <= 1) \text{ then}\)
18: \(\text{Output: unknown location};\)
19: \(\text{end if}\)

**Bootstrapping and incremental update** are two scenarios in which the above algorithm can be applied. In the bootstrapping phase \(e.g.,\) when the device is turned on), our system has no knowledge of the user’s previous location. In this case, the search space in the sign database can be large \(e.g.,\) at city level. Afterwards, in the incremental update phase, the system can use the user’s last known location \(i.f\) it is “fresh” enough) to predict the user’s current coarse-grained location, which can be leveraged to significantly reduce the textual sign search space. This leads to higher accuracy for both speech recognition \(as\) the vocabulary becomes smaller) and the localization algorithm (because a sign corresponds to fewer locations). One way to estimate the user’s coarse-grained location is as follows. Let \((x_0, y_0)\) be the user’s previous location measured \(\Delta T\) minutes ago. Let \(V\) and \(D\) be her maximum walking speed and her range of visibility, respectively. Assuming the user only walks, then the location of signs she currently sees, denoted as \((x, y)\), must satisfy \((x - x_0)^2 + (y - y_0)^2 \leq (V \Delta T + D)^2\). Clearly this is the most simple and conservative estimation, which can be significantly improved by considering walking speed \((11)\), direction \((23)\), and other coarse-grained localization techniques by leveraging various sensors. The bootstrapping phase will be triggered manually or automatically, for example, after sensing the user has taken a vehicle or a train \((13)\). We demonstrate the benefits of incremental update in \((5)\).

3.4 Discussions

In this subsection, we discuss some limitations of our system and potential user interface suitable to our system.
Comparison to Fully-Automated Approach. An inherent limitation of our approach is the localization usually takes longer, and it is difficult to derive a bound for the accuracy, because we rely on human users instead of location sensors. Therefore, our system will not replace automated localization methods. Instead, it provides a useful alternative when traditional approaches cannot work due to various reasons (e.g., in use cases described in §4). Notably, our proposal does have one crucial advantage: by asking users to actively participate in the localization process, it can improve users’ space awareness and their recollection of the environment [10].

Map User Interface. In traditional GPS-based mobile maps, the app directly shows user’s location, as shown in Figure 3(a). However, our approach uses neither the distance nor the direction information of the signs so it is difficult to leverage triangulation to pinpoint the user’s exact location. Instead, we propose to directly highlight the textual signs entered by the user so that she can (probably more easily) figure out her location by using the signs as references. The display is updated when the user enters the next token. Between consecutive updates, the map can move slowly based on user’s walking speed and direction, in order to make the location update smoother. Signs not yet entered can also be displayed (but not highlighted) to help user identify them, and enter them by selecting or speech. This makes sign-based navigation very convenient, as shown in Figure 3(b).

Comparison to Offline Maps. Several vendors (e.g., Garmin and Baidu) provide offline maps allowing users to access them without Internet connectivity. However, users still need to use GPS or Wi-Fi fingerprinting to locate themselves on the map. Our work attempts to bridge this gap by making localization completely offline. Also, Baidu Map [2] provides a “search in view” feature allowing users to search POI (Point of Interest) names within the currently displayed area. But it only searches for one POI (e.g., KFC) at a time and shows all its instances, making it not suitable for localization.

Using General Landmarks. Potentially, the landmarks can be extended from textual signs to more general objects (e.g., a particular statue on a street). However, compared to textual signs, these general landmarks provide much more vague description of the location, and using them as location signatures may cause more ambiguities. We will consider this direction in future work.

4. DATA COLLECTION

To demonstrate the feasibility of our proposal, we collected data from two sources: Google street view and OpenStreetMap [5].

Google street view is a feature in Google Maps that provides panoramic views from positions along many streets in the world. We randomly picked 19 outdoor locations in the U.S., Canada, U.K., and two indoor locations. For the AT&T Building, we conducted a real walk and collected the signs. For all other 20 locations, we randomly select spots with street view, “walk” in the street for a short distance (~0.1 mile), and manually record observed textual signs. The results are shown in Table 3. The street view approach allows us to construct high-quality signature database without physically going to the site, and is ideal for crowdsourcing. For example, an interface can be easily added for users to tag the signs in street view images. Some sign images can also be used as CAPTCHAs so that all Internet users can contribute. We do notice a limitation of using street view images from one single repository: some signs are blocked by obstacles so Table 3 only reports a subset of signs.

OpenStreetMap (OSM). We also crawled OSM to obtain names of POI (point of interest) in much larger areas. OSM [5] is a crowdsourcing-based world mapping project involving 200,000 contributors in 2011 [17]. The OSM raw data [6] employs three main data structures to represent map elements: node, way, and area. We built a custom tool that analyzes these data structures and extracts their associated names (if they exist). We assume each POI has a sign corresponding to its name on the map. For a node, the sign’s location is the same as the node’s. For an area, we pick a random node on its boundary as the sign’s location. For ways, we extracted their names but the road signs’ locations are difficult to estimate. One limitation of this synthetic approach is that it only produces a subset of signs whose contents and locations may also deviate from those of the real ones. However, we believe the dataset is sufficient for our feasibility study in §5 in which we study three areas: London metropolitan area (55K signs), New York City (~21K signs), and the state of California (139K signs).

5. PRELIMINARY RESULTS

Using the two data sources described in §4 we show preliminary results to demonstrate the feasibility of our proposal.

5.1 Characterizing Textual Signs

Sign Popularity. Table 1 summarizes the Google street view results. For the vast majority of the 21 locations covering diverse environments, we observed a non-trivial amount of textual signs. Generally, the signs are denser in downtown than in suburb and residential areas. However, we did not find a strong correlation between sign density and city size (except for very large cities). In indoor environments, signs mostly consist of room names and numbers. We believe our system is applicable in most places. Note that obviously temporary signs (e.g., ad banners) and signs that are too general (e.g., traffic signs) are not included in Table 1.

Token Distribution. Next, we characterize tokens in two locations: the state of California and the London metropolitan area. For each location, we examine tokens in POI signs and tokens in road signs, from OSM. This results in four token sets. Figure 4 shows log-log plot of the ranked frequencies of the four sets. The distributions are heavy-tailed. Although a tiny fraction of tokens, which will be ignored by the localization algorithm, are very general, the vast
5.2 Localization Algorithm

We evaluate the performance of the proposed localization algorithm for different levels of speech recognition accuracy (70%, 80% and 90%) and with different numbers of input signs (3 to 5). We show the results for the New York City and London OSM data sets in Figure 6. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

![Figure 6: The accuracy of localization algorithm.](image)

(a) NYC  
(b) London

Figure 6. The accuracy of localization algorithm.

majority of tokens belong to very few signs (most only one). They are therefore ideal for quickly pinpointing the user’s location.

**Signs in a small area.** We show that if the search is confined within a small area, the number of signs decreases significantly. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

5.3 Speech Recognition (SR)

**Methodology.** We adapt one popular implementation of a SR chain called Sphinx [3]. We use its Android version (PocketSphinx) and make use of the acoustic model as it is. In the experiments, we first randomly pick a set of locations in London and NYC. Then, for each location, we collect token sets from the OSM data with different radii (100m~6400m). Each token set has 67,418 tokens for NYC. We take each token set as input and generate a dictionary and language model files, which are then used by one experimental trial. In each experiment, the participant randomly takes a word from the phones-words dictionary and pronounces it one at a time. The software presents 10 potential candidates. If there was a correct answer, we treat it as a successful recognition. Otherwise, the participant repeats the word again. When he gets a success, we record a rank of the correct candidate.

**Results.** The results are potentially the worst case since the participant was a non-native speaker and we use the Sphinx software out-of-box, i.e., without optimizing acoustic model, dictionary, and language model files.

![Figure 7: SR performance: The results are potentially the worst case since the participant was a non-native speaker and we use the Sphinx software out-of-box, i.e., without optimizing acoustic model, dictionary, and language model files.](image)

(a) Accuracy with # of Attempts.  
(b) Rank Distribution.

Figure 7. SR performance: The results are potentially the worst case since the participant was a non-native speaker and we use the Sphinx software out-of-box, i.e., without optimizing acoustic model, dictionary, and language model files.

the performance of localization. Second, more input signs can lead to better localization accuracy. Even when the speech recognition accuracy is 70%, the proposed algorithm works for more than 90% locations with 4 signs and more than 99% locations with 7 signs.

**Localization Algorithm**

We evaluate the performance of the proposed localization algorithm for different levels of speech recognition accuracy (70%, 80% and 90%) and with different numbers of input signs (3 to 5). We show the results for the New York City and London OSM data sets in Figure 6. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

**Signs in a small area.** We show that if the search is confined within a small area, the number of signs decreases significantly. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

![Figure 6: The accuracy of localization algorithm.](image)

(a) NYC  
(b) London

Figure 6. The accuracy of localization algorithm.

majority of tokens belong to very few signs (most only one). They are therefore ideal for quickly pinpointing the user’s location.

**Signs in a small area.** We show that if the search is confined within a small area, the number of signs decreases significantly. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

5.2 Localization Algorithm

We evaluate the performance of the proposed localization algorithm for different levels of speech recognition accuracy (70%, 80% and 90%) and with different numbers of input signs (3 to 5). We show the results for the New York City and London OSM data sets in Figure 6. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

**Signs in a small area.** We show that if the search is confined within a small area, the number of signs decreases significantly. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

5.2 Localization Algorithm

We evaluate the performance of the proposed localization algorithm for different levels of speech recognition accuracy (70%, 80% and 90%) and with different numbers of input signs (3 to 5). We show the results for the New York City and London OSM data sets in Figure 6. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

**Signs in a small area.** We show that if the search is confined within a small area, the number of signs decreases significantly. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

5.2 Localization Algorithm

We evaluate the performance of the proposed localization algorithm for different levels of speech recognition accuracy (70%, 80% and 90%) and with different numbers of input signs (3 to 5). We show the results for the New York City and London OSM data sets in Figure 6. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

**Signs in a small area.** We show that if the search is confined within a small area, the number of signs decreases significantly. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

5.2 Localization Algorithm

We evaluate the performance of the proposed localization algorithm for different levels of speech recognition accuracy (70%, 80% and 90%) and with different numbers of input signs (3 to 5). We show the results for the New York City and London OSM data sets in Figure 6. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.

**Signs in a small area.** We show that if the search is confined within a small area, the number of signs decreases significantly. We randomly pick 50,000 locations in London. For each location \((x_i, y_i)\), we count the number of signs within \((x_i - x)^2 + (y_i - y)^2 \leq R^2\) where the radius \(R\) ranges from 100m to 6.4km. Figure 5 shows that the number of signs decreases accordingly as the search area becomes smaller. Note that in the incremental update phase, which is the common usage scenario compared to the bootstrapping phase (3.3), usually the search area is small.
has sizes 203 bytes~8 kbytes (100m, 6400m respectively) and the size of the language model is 703 bytes~15 kbytes. In case of the London dataset, the dictionary size is 701 bytes ~ 31 kbytes and the language model size lies in 1.6 kbytes ~ 63 kbytes.

We next examine the recognition performance of SR shown in Figure 7. It is worth mentioning that the results can be the worst case performance since the participant for the experiment was a non-native English speaker and we use the recognition software out-of-box i.e., we did not tune the default acoustic model or the dictionary/language model generated by PocketSphinx. Overall we think the results are encouraging. Up to 400m radius, repeating once more i.e., 2 trials in total, gives 95% of recognition rate. Even for all other cases, 4 trials will cover 90% of accuracy. We note that this is very preliminary result and plan to do a more comprehensive evaluation with a diverse set of participants and more optimized software settings later e.g., using better trained acoustic and dictionary/language models, as well as improved feature acoustic extraction techniques [18].

We also measured the energy consumption of the SR component and compare it with that of GPS sensor. We use Monsoon Power Monitor [4] and Samsung Galaxy S4 device for the measurement. The SR component (including both microphone and SR processing) consumes ~44% less instantaneous power compared to that of GPS (369.22 mW vs. 208.48 mW), which fortifies our use of SR.

6. RELATED WORK

GPS has been used as a primary source of location information and provide accurate data in an outdoor environment. But not all mobile devices are equipped with GPS (Table 1) due to concerns such as energy consumption. Further, GPS sensing does not always provide a precise location in urban environment. It is highly susceptible to obstacles in the sky and can have errors as high as more than 100 meters [16, 21]. Recently a technique for acquiring GPS signal in indoor environment has proposed [19]. However it is not yet applicable to the off-the-shelf devices due to the use of specialized radio components and complexity of the system.

In the last decade, numerous research efforts propose localization techniques based on radio signal based fingerprinting. These include ones based on GSM signal [20], Wi-Fi signal [7], FM signal [8] etc. Unfortunately, some of them are based on the information that is in general not available in off-the-shelf mobile devices [20] or requires heavy training [7]. In contrast, our system uses microphone (or on-screen keyboard) as input, which is mostly available in today’s mobile devices including wearables and our sign database is relatively easy to construct compared to above-mentioned techniques. A number of works aim to strategically combine multiple approaches [21, 15, 25, 14, 22]. They trade-off location accuracy with other system resources, such as energy, thereby different from our approach where the focus is to provide purely local localization systems based on human perception.

When it comes to dealing with textual signs, one intuitive approach is to use image-based positioning [24] i.e., let the device see what a human sees. Although the approach is complementary to our system, it requires multi-stage CV algorithms and large databases (e.g., 300K images for 50K locations at NYC [24]), making cloud the best place for performing such tasks. It thus requires Internet connectivity and brings additional concerns of delay [13], energy consumption, and cellular data usage. Light and viewing angle can significantly reduce the image matching accuracy.

Although explored in slightly different context, the idea of leveraging human assistance to determine a location had been explored and inspired our work. In the GUIDE tourist guide system [9], a visitor is shown several thumbnail images of local attractions and asked if any of them matches the visitor’s surroundings. This approach is not directly applicable to a general-purpose localization system like ours due to scalability issues.

7. CONCLUSION AND FUTURE PLAN

As discussed in [4] there have been a large number of indoor and outdoor localization proposals using various sensing techniques. Instead, we provide a solution for offline localization without using GPS, Wi-Fi, cellular, or Internet connectivity, by exploiting human users’ perception of nearby textual signs. It enables several important use cases, such as offline localization on wearable devices, in both outdoor and indoor environments. Currently we are working on (i) prototyping on Google Glass using speech recognition as the user interface, (ii) building a city-level sign database by tagging Google street view images using crowdsourcing, and (iii) improving speech recognition accuracy, before conducting field trials.

8. REFERENCES