The following publication Z. Peng, S. Gao, B. Xiao, S. Guo and Y. Yang, "CrowdGIS: Updating Digital Maps via Mobile Crowdsensing," in IEEE Transactions on Automation Science and Engineering, vol. 15, no. 1, pp. 369-380, Jan. 2018 is available at https://doi.org/10.1109/TASE.2017.2761793.

CrowdGIS: Updating Digital Maps via Mobile Crowdsensing

Zhe Peng, Shang Gao, Bin Xiao, *Senior Member, IEEE,* Songtao Guo, *Member, IEEE,* and Yuanyuan Yang, *Fellow, IEEE*

***Abstract*—Accurate digital maps play a crucial role in various location-based services and applications. However, store information is usually missing or outdated in current maps. In this paper, we propose CrowdGIS, an automatic store self-updating system for digital maps that leverages street views and sensing data crowdsourced from mobile users. We first develop a new weighted artificial neural network to learn the underlying relationship between estimated positions and real positions to localize user's shooting positions. Then, a novel text detection method is designed by considering two valuable features, including the color and texture information of letters. In this way, we can recognize complete store name instead of individual letters as in the previous study. Furthermore, we transfer the shooting position to the location of recognized stores in the map. Finally, CrowdGIS considers three updating categories (replacing, adding, and deleting) to update changed stores in the map based on the kernel density estimate model. We implement CrowdGIS and conduct extensive experiments in a real outdoor region for 1 month. The evaluation results demonstrate that CrowdGIS effectively accommodates store variations and updates stores to maintain an up-to-date map with high accuracy.**

***Note to Practitioners*—This paper was motivated by the problem of automatically updating digital maps in a manner of mobile crowdsensing. Existing approaches can update stores in maps through a manual survey or update roads automatically from mobile crowdsensing data. Since the store information is a crucial component in digital map, this paper suggests a novel approach to automatically updating stores in digital maps through mobile crowdsensing. This is necessary, in general, because the accuracy of digital map will directly affect the quality of various location based services. Therefore, the system proposed in this paper is useful for engineers and developers to obtain precise digital maps for localization, navigation, automatic drive, etc.**

***Index Terms*—Digital map update, mobile crowdsensing.**

I. Introduction

THE proliferation of mobile computing has prompted the development of map construction techniques based on mobile crowdsourcing. An accurate map with abundant store information can provide users efficient location-based services, including localization, navigation and information sharing. However, stores may be changed and update information may not be available in current maps. Based on real investigations, current digital maps still lack a large amount of store information. Moreover, replaced stores and nonexistent stores cannot be timely updated in the map neither. For example, in some large cities around the world such as Hong Kong, Tokyo and New York, we found plenty of stores are not labelled in the digital map and changed very frequently. Existing inaccurate and out-of-date maps may misguide users and even bring dangers. Therefore, it is crucial to automatically update stores in maps (i.e. replace old stores, add newly-built stores and delete nonexistent stores) to provide better location-based services.

Recently, many works have been devoted to reconstruct and update maps. These works can be classified into three types. The first type is to reconstruct maps based on simultaneous localization and mapping (SLAM) [1][2]. ORB-SLAM2 [3] utilizes monocular, stereo, and RGB-D sensors to perform relocalization, loop closing, and reuse its map in real time on standard CPUs. Authors of [4] present an architecture, protocol, and parallel algorithms for collaborative 3D mapping in the cloud with low-cost robots. The robots run a dense visual odometry algorithm on a smartphone-class processor. The second type updates maps through a manual survey. The state-of-the-art Google map leverages crowdsourcing for map updating, where user-submitted changes are integrated into their map after a manual review. The third type is to automatically update maps. CrowdAtlas [5] automate road updating in a map based on people's travels, either individually or crowdsourced. It uses a mobile navigation app to detect significant portions of GPS traces that do not conform to the existing map. Roads are updated in the map when sufficient traces are collected. AcMu [6] updates WiFi Received Signal Strength (RSS) of each position in a map for wireless indoor localization. By accurately pinpointing mobile devices, the system can collect real-time RSS samples when devices are static. With these reference data, the system updates the complete radio map by learning an underlying relationship of RSS dependency between different locations.

While existing studies have tried to explore the possibility of updating map, accurate store update in a digital map deserves more attention. First, SLAM-based approaches mainly focus on reconstructing maps, which cannot be directly utilized to find changes and update maps. And these methods often

Z. Peng, and S. Gao are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.
E-mail: {cszpeng, cssgao}@comp.polyu.edu.hk.

B. Xiao is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, and The Hong Kong Polytechnic University Shenzhen Research Institute, Shenzhen 518000, P.R. China.
E-mail: csbxiao@comp.polyu.edu.hk.

S. Guo is with the College of Electronic and Information Engineering, Southwest University, Chongqing 400715, P.R. China.
E-mail: stguo@swu.edu.cn.

Y. Yang is with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook NY 11794, USA.
E-mail: yuanyuan.yang@stonybrook.edu.

Manuscript received May 8, 2017; revised September 4, 2017; accepted October 3, 2017.

need extra devices, such as depth camera, robot and vehicle. Second, updating stores through manual reviews as Google is both effort-intensive and time-consuming [7]. Even though the store owner can actively apply to update his store, such information is not sufficient and changed stores may not be updated in time, especially for nonexistent stores. Third, previous works can update specific components in maps. For example, crowdAtlas [5] focuses on updating changed roads in maps from GPS traces collected via crowdsourcing. AcMu [6] mainly updates WiFi RSS values of different positions in maps from sensor data. However, in a real situation, besides roads and WiFi RSSs, stores also need to be updated, such as replacing old stores, adding newly-built stores and deleting nonexistent stores. The names and positions of stores in maps are extremely valuable information for user's reference. Third, these studies heavily rely on sensory data. Except sensory information, visual information can preserve more context information for an unknown environment, such as the geometric information, color information, and text information. Visual information based approaches may provide more accurate geometric (shape, coordinates and orientations) information compared with the sensor-only approaches. As a consequence, our further research problem would be: *Can we provide a practical and effective approach to automatically update stores in a digital map through mobile crowdsensing?*

In this paper, we propose an affirmative answer through the systematic design and implementation of *CrowdGIS*, which enables stores to be automatically replaced, added, and deleted in maps leveraging mobile crowdsourced data. Different from GPS-based schemes, we estimate user's shooting positions from both GPS and images by proposed joint position estimation scheme. Specially, an underlying relationship between estimated positions and real positions is learned through developed new weighted artificial neural network. Then, a novel text detection method is designed by considering two valuable features to recognize complete store name instead of individual letters as previous study. After that, we transfer the shooting position to store position in the map. According to real observations, we further consider three various categories of updating stores: replacing old stores, adding newly-built stores and deleting nonexistent stores. To accurately localize and update changed stores, position estimation method is proposed based on the kernel density estimate model. CrowdGIS can save extensive manpower and time to effectively update stores in a digital map. When more stores are updated, users can locate their positions more precisely and receive much better location-based services. To the best of our knowledge, our work represents the first attempt to cope with store variations to automatically update stores via mobile crowdsourcing.

The automatical store update requires store localization and recognition from street views taken from smartphones. Thus, implementing such a functional system entails distinct challenges. (1) Localizing shooting positions with high accuracy from street views and sensing data. (2) Precisely recognizing various stores from street views. (3) Accurately localizing the stores recognized from street views in the map. (4) Updating changed stores in the map based on their various categories.

To address the above challenges, we make the following contributions in this work:

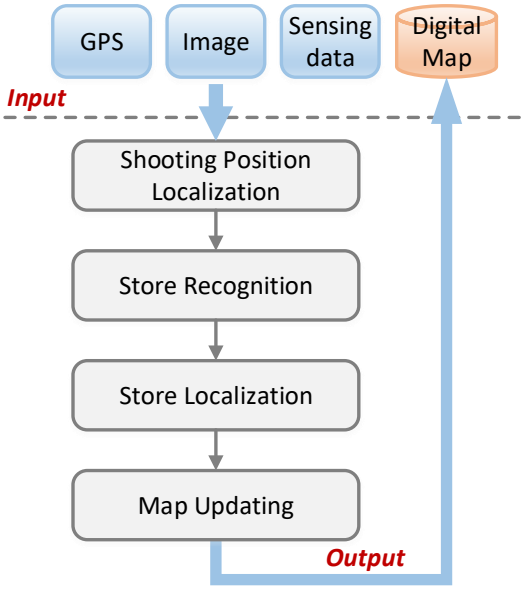


Fig. 1: The architecture of CrowdGIS system.

- We propose the *CrowdGIS* system architecture which leverages mobile crowdsourced data to automatically update stores in a digital map.
- Different from previous GPS-based schemes, we localize user's positions from both GPS and images. An underlying relationship between various estimated positions and the real position is learned to accurately localize user's shooting positions. The average localization accuracy can be improved by about 25%.
- A novel store name recognition method is proposed by considering two valuable features (i.e. the position of text in image and the colour histogram). In this way, we are able to recognize the complete store name instead of individual letters as previous study. The results show that about 80% store names can be accurately recognized.
- According to real observations, we consider three various categories of updating stores: replacing old stores, adding newly-built stores and deleting nonexistent stores. To accurately localize changed stores, we estimate their positions based on the kernel density estimate model. More than 75% stores can be updated correctly, and an average accuracy of about 8 meters can be achieved.
- In addition, we develop a prototype and conduct extensive evaluations in a real outdoor region for 1 month. The results illustrate that CrowdGIS effectively accommodates store variations and maintains an up-to-date map.

The remainder of this paper is organized as follows. We first describe system overview in Section II and provide preliminary techniques in Section III. Then we detail system design in Section IV and present system implementations and evaluations in Section V. We review related works in Section VI and conclude our work in Section VII.

II. System Overview

High-performance sensors can provide abundant movement information and street views can offer luxuriant visible and valuable description about surroundings. Accordingly, we propose CrowdGIS system to update stores in the map leveraging street views and sensing data crowdsourced from smartphones. This system consists of four major stages. The CrowdGIS system architecture is sketched in Fig. 1.

Shooting Position Localization. CrowdGIS utilizes street views and sensing data to jointly estimate the shooting positions of captured street views. For each captured street view, we calculate four shooting position candidates by four various methods (i.e. one GPS method and three image matching methods). Then the shooting position is predicted through the relationship between four shooting position candidates and the real shooting position, which is pre-learned from collected training data.

Store Recognition. Considering two types of stores (with or without logos), we integrate two store recognition approaches to recognizing various store names from street views. For stores with logos, we utilize the object detection technique to recognize them from logos. For stores without logos, we propose a novel store name recognition method to divide and extract store names by considering two effective features (distance and colour).

Store Localization. After recognizing store names, CrowdGIS localizes these stores in the map based on the captured street views and corresponding shooting positions. We extract the position of store in street view and then transfer it into the global coordinate system of the map through calculating its shooting direction.

Map Updating. With the obtained names and positions of stores from various street views, we propose a map updating method to update stores in the map. Based on observations in real environment, we classify changed stores into three categories (i.e. replacing old stores, adding newly-built stores and deleting nonexistent stores) and then update them with corresponding approach. Specifically, we design a position estimation model to calculate the accurate position of updating store based on kernel density estimate model.

III. Preliminaries

In this section, we briefly review some techniques behind our system, and clarify their necessity for our purpose.

Pinhole Camera: We use the classical Pinhole Camera [8] to model the imaging principle and photograph parameters acquiring. In this model, 3D points in the real world and their projected points in the image plane construct an ideal pinhole camera, where its aperture is described as a point and no lenses are used for focusing light. In this way, we can ignore geometric distortions and unfocused blurring caused by lenses and finite sized apertures. Based on this model, we can acquire the angle of view, the size of photo from the smartphone's camera parameters, which are essential factors for store recognition and localization.

Artificial Neural Network: Artificial Neural Network (ANN) [9] is a computational multi-layer model based on the structure and functions of interconnected neurons. ANN is utilized to find relationships between inputs and outputs given finite data samples. The expression of the weighted sum to the k-th neuron in the j-th layer ($j \geq 2$) is given by

$$S_{j,k} = \sum_{i=1}^{N_{j-1}} (\omega_{j-1,i,k} I_{j-1,i}) + b_{j,k} \tag{1}$$

where $I_{j-1,i}$ is the information from the i-th neuron in the $(j-1)$-th layer, $b_{j,k}$ is the bias term and N_{j-1} is the number of neurons in the $(j-1)$-th layer. Thus, drawing on the ANN framework, we can find the unknown functions between several estimated positions and real position to accurately localize the shooting position of captured street view.

Manhattan World Assumption: Most man-made scenes follow the Manhattan World Assumption [10], where Cartesian coordinate system is used as a Manhattan grid. All lines in a photo image are assumed parallel to three directions. Accordingly, we can extract text aligned with the three Manhattan directions from street view. Hence various store names can be recognized through clustering extracted letters and numbers.

Kernel Density Estimate: Kernel Density Estimate (KDE) [11] is a data smoothing method used to estimate the probability density function based on a finite data sample set. Let $(x_1, x_2, ..., x_n)$ be an independent and identically distribution samples drawn from some distribution with an unknown density function f. The unknown density function f can be estimated by kernel density estimate (KDE) as following:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K_i(\frac{x - x_i}{h}), \tag{2}$$

where $K_i(\cdot)$ is the kernel function (a nonnegative function that integrates to one and has mean zero) of the sample x_i, and $h > 0$ is a smoothing parameter called bandwidth. Since the store position is unknown, the KDE is a powerful tool to estimate the probability distribution of store position from a finite set of candidate positions in the map.

IV. System Design

In this section, we first illustrate proposed method to localize shooting positions from collected street views and sensing data. Then we present schemes to recognize and localize various stores from captured street views, and further update changed stores in the map.

A. Shooting Position Localization

In this subsection, we propose a novel scheme to localize the shooting position of street views captured by user's smartphones. Traditional methods just utilize GPS data collected by smartphones to localize shooting position. However, the accuracy of these methods is very limited because of the inherent error in GPS sensor. Instead, street views captured by users also provide extremely valuable information about surroundings, which can be utilized to localize shooting position. With this insight, we propose a novel joint position estimation algorithm to localize the shooting position of street

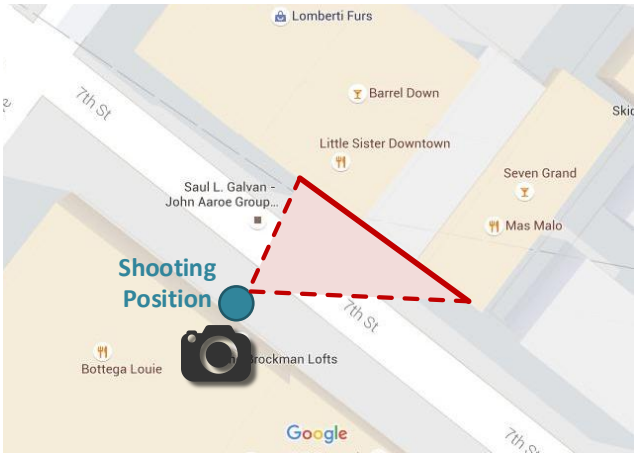


Fig. 2: An example of capturing a street view from shooting position.

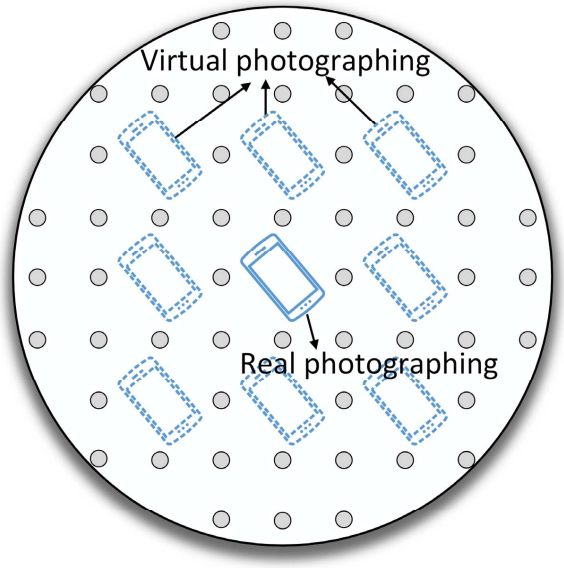


Fig. 3: Virtual photographing in various positions to estimate shooting position from image.

view, combining the GPS data and images. Fig. 2 gives an example of capturing a street view from shooting position.

Shooting position estimation from GPS. We collect a sequence of GPS samples to estimate shooting position. The GPS data contain two parts: one GPS sample while photographing which is called *start data* denoted as s, and a series of GPS samples when the user moves after photographing which are called *tail data* denoted as $\{t_1, t_2, \cdots, t_n\}$. The estimated shooting position from GPS is denoted as $\mathbf{x_G}$. To calculate the position $\mathbf{x_G}$, we utilize a road matching method [12] to match the collected GPS data with the existing Google map. The start data s is calibrated to a new position, which is viewed as the estimated shooting position $\mathbf{x_G}$ from GPS.

Shooting position estimation from image. Besides the GPS data, we also leverage captured street view to estimate shooting position. Through matching captured street view with existing Google street views via three image retrieval algorithms [13][14][15], we acquire three estimated shooting positions donated as $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$.

When a user captures a street view, various parameters of photographing are collected from smartphone. These parameters include the direction of photographing $\overrightarrow{D}$, the angle of pitching ω_p, the angle of view ω_v and the size of photo s_p.

After a street view is captured, CrowdGIS fetches existing Google street views of various positions around the user with the same parameters of photographing as the user. These Google street views are downloaded through the Google Street View API. Since the system simulates photographing as the user in various positions with the same parameters, this process is called *virtual photographing*, as shown in Fig. 3. To narrow the data size of downloaded street views, we obtain one Google street view from every position around the user with interval of 2 $meters$ in a range of 50 $meters$ radius.

Then CrowdGIS estimates shooting position through matching captured street view with the downloaded Google street view set. As an image can be represented by three major categories of features (i.e. colour, texture and shape), we adopt three state-of-the-art image retrieval methods [13][14][15] to match street views. Each method outputs a most similar street view and its corresponding shooting position. Three estimated shooting positions are obtained and donated as $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$.

Specially, if a store is changed and different from the existing street view, our method is still effective. This is because the GPS first gives a rough position and a limited area. Then although a store is changed, the surroundings are generally unchanged (such as building, nearby stores and road signs). Street views captured by crowdsourcing will include these unchanged surroundings. The image matching method only finds the most similar street view. Thus, captured store can be matched and localized, even if it has been changed.

Joint position estimation of shooting position. To accurately localize shooting position from four estimated shooting positions, we propose a novel joint location estimation algorithm based on artificial neural network. Given several candidate shooting positions estimated from GPS and images, a naive method is directly taking the average position of all candidates as the shooting position. However, the average position may not be accurate since this method supposes a linear relationship between the candidates and real shooting position, which is impractical and problematic. Alternatively, artificial neural network (ANN) [9] is a superior choice to learn the unknown function between the candidates and real shooting position. Moreover, since the contributions of estimated positions are not previously known, we propose a weighted ANN to localize shooting positions.

First, we calculate the accuracy of each candidate shooting position. For the shooting position $\mathbf{x_G}$ estimated from GPS, this position is calibrated from the *start data* s with a calibration distance d. Although the distribution of GPS location is not perfectly Gaussian because of the shape and acceleration of satellites and the atmosphere turbulence, its error can be estimated and bounded by a Gaussian distribution [16]. We

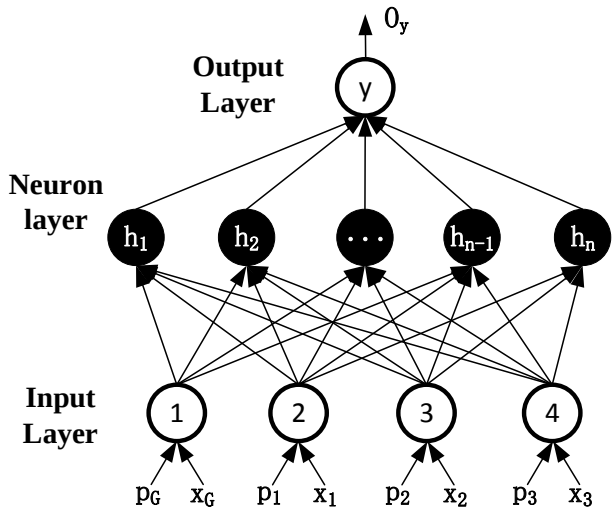
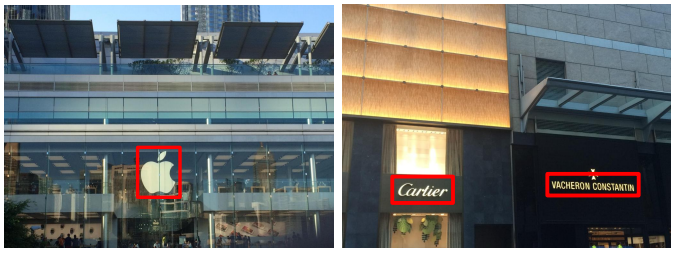


Fig. 4: Weighted artificial neural network architecture.



(a) Recognize store name from logo. (b) Recognize store name from text.

Fig. 5: An example of store recognition.

have the distribution of calibration distance d:

$$f(d) = \frac{1}{\sqrt{2\pi}\sigma_G} e^{-\frac{d^2}{2\sigma_G^2}} \tag{3}$$

with the standard deviation denoted as σ_G. Thus, the accuracy $p_G \in [0,1]$ of estimated shooting position $\mathbf{x_G}$ is derived from the distribution $f(d)$ of calibration distance d.

For the shooting positions ($\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$) estimated from images, the corresponding accuracy p_1, p_2 and p_3, $p_{1,2,3} \in [0,1]$ is defined from the Hamming Distance H [17] of two matched street views as follows:

$$p_i = \frac{H_i}{s_p^i}, \tag{4}$$

where s_p^i is the size of matched street view for shooting position $\mathbf{x}_i$.

Second, we design a weighted artificial neural network to learn the relationship between four estimated shooting positions and real shooting position. In our case, since each estimated shooting position contributes to the final estimation differently, we design the input $\boldsymbol{I}$ of artificial neural network as weighted estimated shooting positions $\boldsymbol{I} = \{p_G\mathbf{x_G}, p_1\mathbf{x}_1, p_2\mathbf{x}_2, p_3\mathbf{x}_3\}$. With these weights, more accurate relationship function can be learned. The architecture of weighted artificial neural network is shown in Fig. 4. The output of the kth neuron in the neuron layer is

$$O_k = \frac{1}{1 + exp(-S_k)}, \tag{5}$$

where S_k is calculated from Equation (1). And the objective function F of neural network is:

$$F = \frac{1}{2} \sum_{i=1}^{N_d} \sum_{s=1}^{N_L} (X_s(i) - O_s(i))^2, \tag{6}$$

where N_d is the number of examples in the data set, N_L corresponds to the number of outputs of the neural network, X_s represents the target value corresponding to the sth neuron of the output layer. In our system, the output of weighted Neural Network is the real shooting position. After the training process, the relationship between estimated shooting positions and real shooting position can be learned. Thus, utilizing the relationship learned from the weighted ANN, CrowdGIS localizes the shooting position from estimated shooting positions.

B. Store Recognition

In this subsection, we recognize various store names from captured street views. For stores with logos, we utilize object detection technique to recognize their names. For stores without logos, we propose a text clustering method to divide and extract various store names.

Store name recognition from logo. Motivated by the observation that most stores have their unique logos, especially for chain stores, we incorporate object detection technique to recognize stores with logos. Fig. 5 (a) gives an example. For each captured street view, we first extract the histogram of oriented gradients (HOG) and scale-invariant feature transform (SIFT) features to represent image information. Then we utilize the Locality-constrained Linear Coding (LLC) [18] to further encode the local features into final image histograms. The LLC utilizes the locality constraints to project each descriptor into its local-coordinate system and captures the correlations between similar local features by sharing the visual words, thus it could give better detection results compared with the traditional bag of visual words methods. For the image classification, we apply the Multiple Kernel Boosting (MKB) [19] to classify the logos. MKB is a boosted Multiple Kernel Learning (MKL) method, which combines several SVMs of different kernels [20], [21], thus it could provide better classification performance. Thus, through recognizing corresponding logos, the names of stores with logos are obtained.

Store name recognition from text. Existing text recognition methods could recognize letters, numbers and words from an image with extremely high accuracy. However, in many cases, the store name is a non-semantic word (such as "GEOX") or a combination of words (such as "bread n butter"). Moreover, the captured street view may include several stores. Hence, to accurately recognize stores without logos, we propose a text clustering technique to infer various store names based on the Manhattan World Assumption.

With the observation that most texts of store names are aligned within the three Manhattan directions, we propose a method to identify various store names from captured street

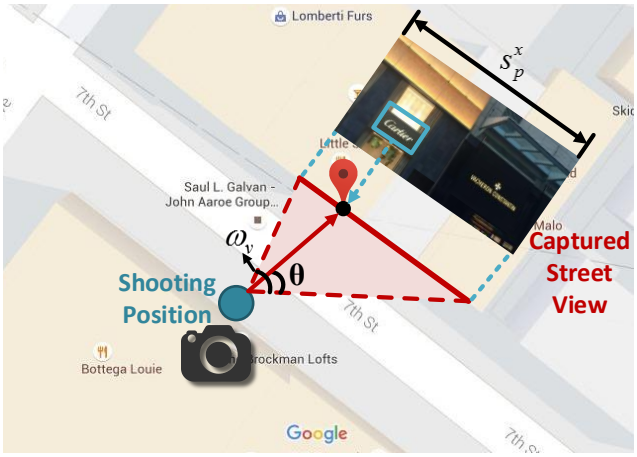


Fig. 6: Localizing a store in the map through calculating its deflection angle θ.

views. Fig. 5 (b) gives an example. After recognizing all letters and numbers from street view with text recognition [22] technique, we characterize each letter and number with two features: the position in street view image and color histogram. Then we cluster these letters and numbers to separate various store names with BIRCH [23] method. Thus, letters and numbers with close distance and similar colour histogram are clustered into one group to generate a store name.

C. Store Localization

In this subsection, we illustrate the method to localize stores in the map based on the captured street views and corresponding shooting positions. We extract the position of store in the captured street view and then transfer it into the global coordinate system of the map through calculating its shooting direction.

As shown in Fig. 6, a recognized store name is bounded with a box. We define the center position of bounding box as the position (x_{local}, y_{local}) of recognized store in the local coordinate system of captured street view. According to the principle of pinhole imaging, the deflection angle θ of recognized store in the global coordinate system is:

$$\theta = \omega_v - \frac{x_{local}}{s_p^x} \times \omega_v, \tag{7}$$

where ω_v is the angle of view and s_p^x is the length size of photo. Since the direction of photographing $\overrightarrow{D}$ collected from smartphone corresponds to half of the angle of view (i.e. $\frac{\omega_v}{2}$), the shooting direction of recognized store is obtained.

Because users capture street views within their sights, the behind concealed stores can not be captured. Thus, with the given shooting position and the angle of pitching ω_p, recognized store in the global coordinate system of the map is localized at the first intersection position of its shooting direction and walls in the Google map.

D. Map Updating

We update the maps with the obtained names and positions of stores from street views. Changed stores are classified into three categories, and updated with corresponding approach. Specifically, we design a position estimation model to calculate the position of updating store based on kernel density estimate.

Algorithm 1: Classifying three updating categories

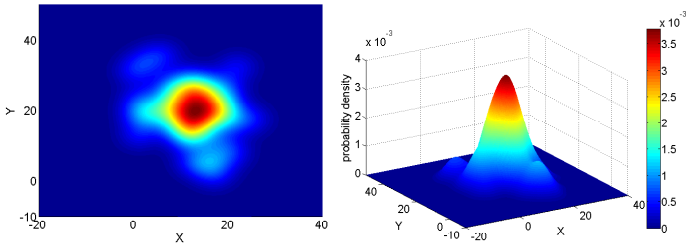
Input:
store tuple set from street view $\mathbf{S} = \{< n_s^i, p_s^i >\}$;
store tuple set from map $\mathbf{M} = \{< n_m^j, p_m^j >\}$;
Output:
category indicator vector Λ;

for *all* (S_i, M_j) **do**
  initialize Λ with $null$;
  if n_s^i == n_m^j *and* $\|p_s^i - p_m^j\| \leq \varepsilon$ **then**
    $\Lambda \leftarrow maintain$;
    remove the stores from tuple set **S** and **M**;
  end
  if n_s^i != n_m^j *and* $\|p_s^i - p_m^j\| \leq \varepsilon$ **then**
    $\Lambda \leftarrow replace$;
    remove the stores from tuple set **S** and **M**;
  end
end
for *all stores left in set* ***S*** **do**
  $\Lambda \leftarrow add$;
end
for *all stores left in set* ***M*** **do**
  $\Lambda \leftarrow delete$;
end

Defining three updating categories. Street views captured by smartphones provide reliable information about the changes of stores in the map. Based on observations in real environments, we consider three categories of updating stores: *replacing* old stores, *adding* newly-built stores and *deleting* nonexistent stores.

We classify changed stores into three updating categories by comparing their names and positions with that in the map. The process is described in Algorithm 1. Stores identified from street views are denoted as a set of tuple $\mathbf{S} = \{< n_s^i, p_s^i >\}, i = 1, 2, 3...$, where n_s^i represents the name of store i and p_s^i represents the position of store i, and stores in the existing Google map are denoted as a set of tuple $\mathbf{M} = \{< n_m^j, p_m^j >\}, j = 1, 2, 3...$, where n_m^j represents the name of store j and p_m^j represents the position of store j. For each store in both sets, we set an indicator Λ to indicate the category of updating it belongs to. We compare the name and position of each store in both sets. If the names of two stores are the same and their positions are extremely close, it proves that the store is unchanged in the map. If two store names are different in the same position, the store in the map may be replaced. For the stores identified from street views, if they cannot be matched with the map, we classify they in the newly-built store category. In contrast, if there exist unmatched stores in the map, they are viewed as potential nonexistent stores. Considering errors in the accuracy of position, we think two positions are the same if their distance is less than ε.

Updating stores in the map. To improve the robustness of our system, a significant principle for updating a store is that



(a) The 2D view of KDE. (b) The 3D view of KDE.

Fig. 7: Estimating probability density function of store position based on KDE.

this store has been photographed and indicated to conduct the same updating operation for sufficient times. Thus, for each category of updating stores, if a store has been indicated to conduct the same updating operation for sufficient times (more than a presupposed *support threshold*), the system conducts corresponding updating operation.

Deleting nonexistent stores. If a store in current map have not been captured in nearby street views for more than presupposed times, this store is viewed as nonexistent and deleted from current map. Because a large number of street views are collected through crowdsourcing, it is rational to assume that real existing stores can be captured in various street views.

Adding newly-built stores. If a store is indicated to exist in a limited arrange for more than presupposed times, this store is viewed as newly-built and added in current map. Because indicated positions may not be exactly same in each time, we estimate its precise position based on Kernel Density Estimate (KDE) [11]. We model the distribution of the i-th indicated position of store $\mathbf{A}$ as a normal distribution. The probability density function in each position $\mathbf{x}$ is:

$$f_i(\mathbf{x}) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{\|\mathbf{x}-\mu_i\|^2}{2\sigma_i^2}}, \tag{8}$$

where μ_i is the position of i-th candidate of store $\mathbf{A}$, σ_i is the standard deviation of the position. Considering the probability distributions of every indicated position are mutually independent, we model the total probability density function $\hat{f}$ by using KDE:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh} \sum_{i}^{n} f_i(\frac{\mathbf{x}}{h}). \tag{9}$$

According to Equation (9), we estimate the position of store $\mathbf{A}$ as the position $\mathbf{x}$ which owns the highest $\hat{f}(\mathbf{x})$. Fig. 7 gives an example that probability density function of store position is estimated based on KDE.

Replacing old stores. Similarly, if a store in current map has been indicated to change into another store for more than presupposed times, this store is viewed as out-of-date. The progress of replacing a store can be considered as a combination of deleting and adding a store.

Thus, through frequently and timely updating, the map is almost always up-to-date and gracefully adapts to real environment changes.

V. Implementations and Evaluation

A. Experimental Methodology

We implement CrowdGIS on an Intel core i7 machine with 64GB RAM and NVIDIA TITAN X graphics card. The high-performance graphics card supports sensor data and image analysis in this work. We conduct experiments in an outdoor region covering about 3,000m×3,000m in Hong Kong. Specifically, the experiment area belongs to an urban environment, which contains 261 various stores. The stores include convenience stores, supermarkets, banks, etc.

We recruit ten volunteers to collect street views and sensing data for 1 month. Each user carries a smartphone during his daily life. The smartphones are pre-installed with a developed APP for automatically collecting sensing data while the user captures street views. When the users travel in the experiment area, they capture street views as they commonly do. The users do not need to behave intentionally for the sensing data collection. We believe that the data gathered in such way are representative for general realistic scenarios.

Besides the street views, various sensing data are automatically collected from sensors, which include GPS, accelerometer, gyroscope, and compass. The users capture one or multiple street views in one position and then continue to walk. During this process, the sensing data collection procedure is triggered for a certain period (ranging from five seconds to five minutes). The collected sensing data record the orientation and position of photographing and the subsequent moving trajectory. In our evaluation, we collect 8,471 images and 2,615 moving trajectories from volunteers.

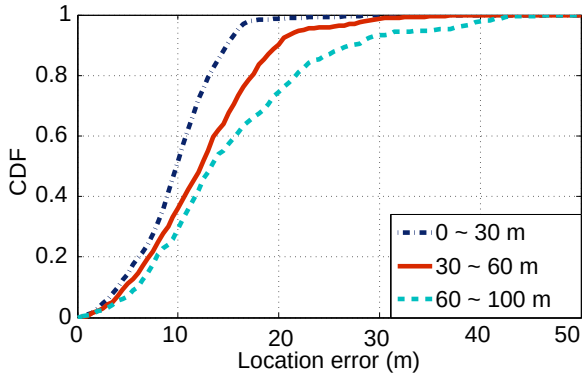
To evaluate the location error of our system, we also collect real shooting positions of street views captured by the volunteers. For the weighted ANN utilized in shooting position localization, we build the training data set through sampling 1000 various positions in an outdoor area (different form the experiment area). And we set the number of neurons as 12 to achieve the best training performance after comparing with other neuron numbers.

OpenCV library (version 3.0) is adopted to identify stores from logos. In particular, a training data (logo images) set is built in advance to conduct the image recognition. The training images are collected from two parts, which contains 2,000 images about 100 various stores. The first part images (300 out of 2000 samples) are collected via photographing logos from various viewpoints in the real situation (different form the experiment area). The second part images (1700 out of 2000 samples) are downloaded from the internet to enable the generality of the images.

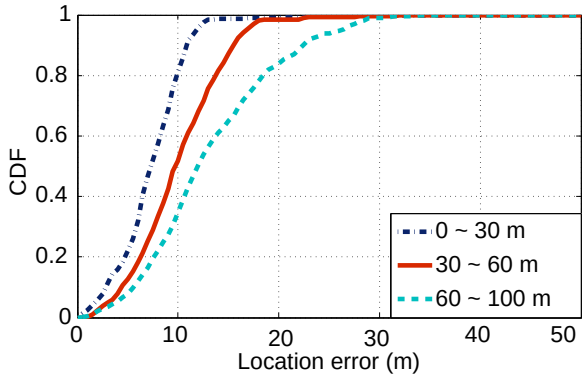
B. Performance Evaluation

Since CrowdGIS consists of four key modules, we evaluate each module to better understand the effectiveness and limitation of system.

Performance of street view localization. We first evaluate the localization performance of the proposed street view localization algorithm. Since street views are utilized to improve the accuracy of localization, we compare the performance of localization with and without image data. As shown in Fig. 8

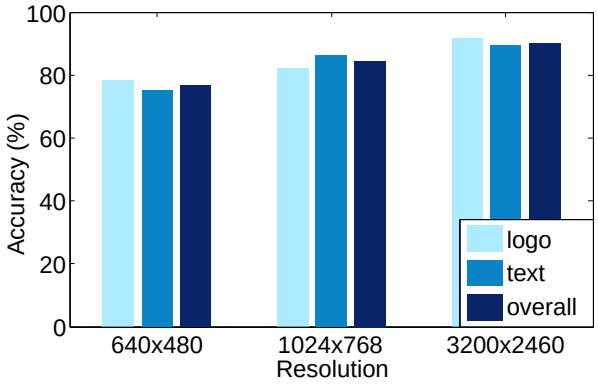


(a) Position estimation from GPS.

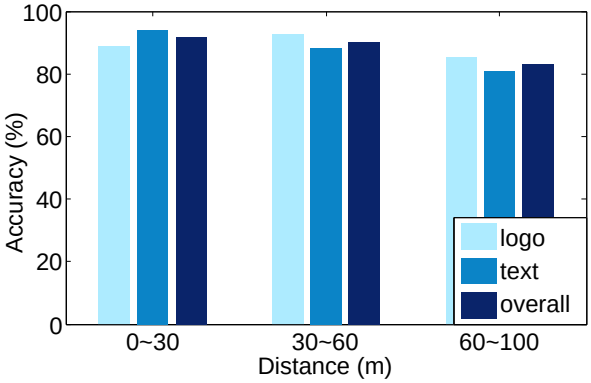


(b) Joint position estimation from GPS and image.

Fig. 8: Performance of street view localization.



(a) Accuracy of identification vs. resolution.



(b) Accuracy of identification vs. distance.

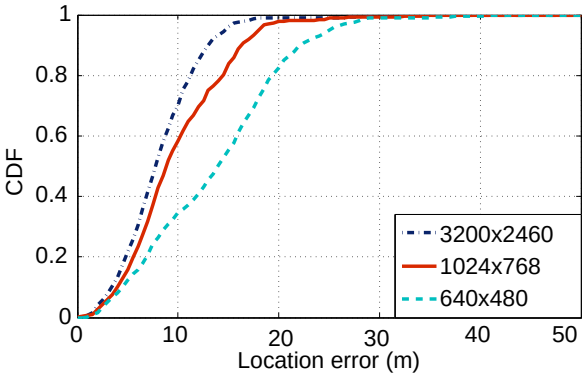
Fig. 9: Performance of store identification.

(a), localizing street views from GPS yields average accuracy of about 10.5 meters and 90th percentile accuracy of 15.7 meters when the distance between user and street view is less than 30 meters. An average accuracy of about 12.4 meters and 90th percentile accuracy of 20.2 meters are achieved when the distance is between 30 meters and 60 meters. The location accuracy degrades to 18.2 meters in average error and 27.4 meters in 90th percentile error when the distance is more than 60 meters. The results show that street view localization has less accuracy when user stands farther away from street view.

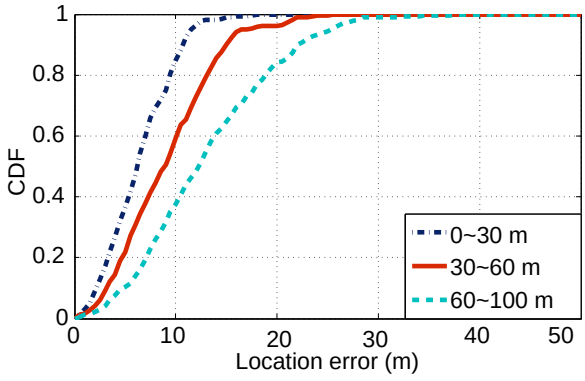
In contrast, Fig. 8 (b) illustrates better street view localization performance when CrowdGIS combines GPS data and street views. CrowdGIS localizes street views with the average accuracy of about 7.3 meters and 90th percentile accuracy of 11.6 meters when the distance between user and street view is less than 30 meters. When the distance is between 30 meters and 60 meters, street view localization yields 9.4 meters in average error and 15.3 meters in 90th percentile error. And the location accuracy degrades to 12.1 meters in average error and 22.1 meters in 90th percentile error when the distance is more than 60 meters. The high accuracy is benefitted from the stable performance of joint position estimation combining GPS and image data.

Performance of store identification. Precision of the store identification is a critical criteria of CrowdGIS. We evaluate the performance with various image resolutions and photographing distances. Fig. 9 (a) shows the accuracy of store identification with different image resolutions. When the captured street view resolution is 640×480, store identification has an accuracy of 78.3% from logo, 75.2% from text and overall 76.9%. When the resolution is 1024×768, the accuracy is 82.1% from logo, 86.2% from text and overall 84.5%. And when the resolution is 3200×2460, CrowdGIS achieves better accuracy with 91.5% from logo, 89.4% from text and overall 90.2%. Fig. 9 (b) shows the accuracy of store identification with different photographing distances. When the distance between user and street view is less than 30 meters, store identification attains an accuracy of 89.1% from logo, 93.8% from text and overall 91.9%. When the distance is between 30 meters and 60 meters, the accuracy is 92.8% from logo, 88.1% from text and overall 90.1%. And when the distance is more than 60 meters, the accuracy degrades to 85.2% from logo, 81.1% from text and overall 82.7%. Thus, with various image resolutions and photographing distances, CrowdGIS accurately identifies various stores from captured street views.

Performance of store localization. We use location error to evaluate the performance of store localization. As shown in Fig. 10 (a), CrowdGIS produces average accuracy of 8.4 meters and 90th percentile accuracy of 12.2 meters with resolution 3200×2460. An average accuracy of 9.1 meters and 90th percentile accuracy of 16.1 meters are achieved with resolution 1024×768. The location accuracy degrades to 14.4 meters in average error and 22.6 meters in 90th percentile error with lower resolution 640×480. Fig. 10 (b) shows the performance with various photographing distances.

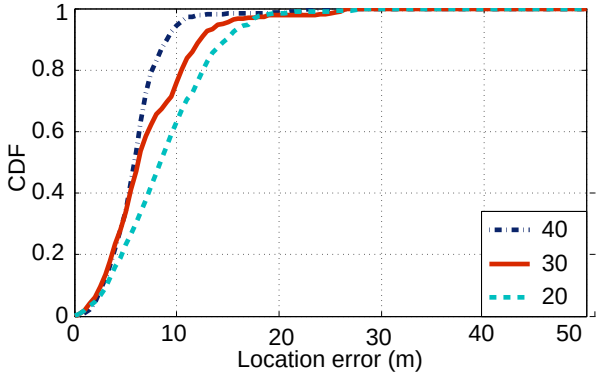


(a) Accuracy of localization vs. resolution.

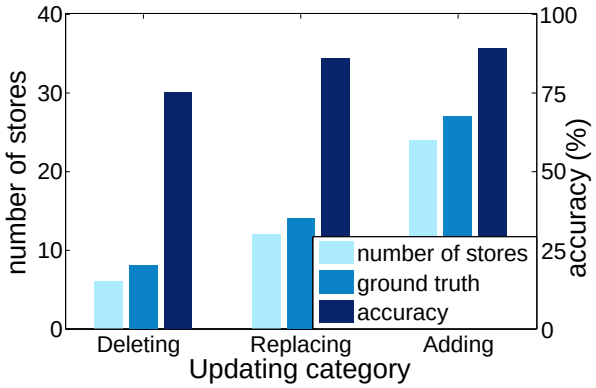


(b) Accuracy of localization vs. distance.

Fig. 10: Performance of store localization.



(a) Position accuracy vs. support threshold.



(b) Name accuracy vs. updating category.

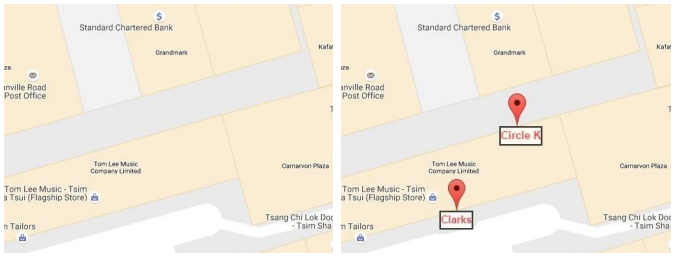
Fig. 11: Performance of map updating.

Store localization yields average accuracy of 8.1 meters and 90th percentile accuracy of 11.1 meters when the distance between user and street view is less than 30 meters. An average accuracy of 8.8 meters and 90th percentile accuracy of 14.8 meters are achieved when the distance is between 30 meters and 60 meters. The location accuracy degrades to 12.5 meters in average error and 24.3 meters in 90th percentile error when the distance is more than 60 meters. Therefore in practice, CrowdGIS localizes stores from captured street views and collected sensing data with high accuracy.

Performance of map updating. We evaluate the performance of map updating through two aspects: the accuracy of updated store position and the accuracy of updated store name. Since updating a store in the map is triggered by collecting sufficient number of candidates, we conduct experiments with different support thresholds. Fig. 11 (a) shows the relationship between the accuracy of updated store position and support threshold. CrowdGIS achieves average accuracy of about 9.1 meters and 90th percentile accuracy of 15.2 meters when the support threshold is set to 20. If we set the support threshold to 30, the average accuracy is 6.3 meters and 90th percentile accuracy is 11.9 meters. A better performance is obtained with average accuracy of 6.1 meters and 90th percentile accuracy of 9.8 meters when the support threshold is added to 40.

Moreover, Fig. 11 (b) shows the accuracy of updating store names in three categories. For the deleting category, 8 stores are nonexistent in the experiment area and CrowdGIS successfully detects and deletes 6 stores , which achieves 75% accuracy. For the replacing category, there are 14 stores are changed to other stores. CrowdGIS correctly recognizes 12 stores and replaces them with new stores, which achieves 85.7% accuracy. For the adding category, some stores are not labelled in current map and some stores are new-built, thus we find 27 stores should be added into the map. CrowdGIS succeeds in adding 24 stores into the map with 88.9% accuracy. Fig. 12 and Fig. 13 give some examples of updating changed stores in the map. These experiment results demonstrate that utilizing data collected by only four volunteer users in 1 month, CrowdGIS achieves comparable accuracy and quantity of updating stores in the map. In other words, maps can be continuously updated to maintain their accuracy when abundance crowdsourcing data are collected. Thus, we envision CrowdGIS as a fundamental and indispensable reviser for existing maps to cope with store variations.

Limitation of the proposed CrowdGIS. Although the proposed CrowdGIS performs favorably against existing methods in updating digital maps with higher accuracy, it is far from perfect. It might make mistakes or give wrong information in certain cases as shown in Fig. 14. The proposed method does not work well if a store name is designed unusually [Fig. 14 (a) and Fig. 14 (b)], and the store name is hard to be recognized correctly. In such case, a store may be added in the map with a wrong name. In addition, when a store name is covered by an obstacle, the proposed method does not work well [Fig. 14



(a) The existing map. (b) Two stores are added in the map.

Fig. 12: An example of adding stores.

(a) The existing map. (b) Stores are replaced and deleted.

Fig. 13: An example of replacing and deleting stores.

(c) and Fig. 14 (d)] and the store may be deleted in the map.

VI. Related Work

Among a large body of works in the literature of map updating, the design of CrowdGIS is closely related to the following categories of research.

Map Construction and Updating. Smartphones with various built-in sensors [24], [25] have been thoroughly exploited to construct maps. Recent innovations such as Walkie-Markie [26] propose to construct indoor maps through detecting various landmarks. Such landmarks can also be used to localize users and calibrate users' trajectories in the localization systems [27], [28], [29]. CrowdInside [30] is a crowdsourcing-based system for automatically constructing maps. It leverages various inertial sensing data collected from the smartphones to generate user motion traces. As these works mainly aim at constructing maps in the initializing phase, CrowdGIS is orthogonal to them in focusing on updating existing maps during serving phase to cope with store variations over time. Thus, CrowdGIS is compatible to the map construction by using schemes in these works.

Considering environmental dynamics, LEMT [31] utilizes reference points to learn a relationship of one location and its neighbors to cope with the RSS variations. Radio map can be updated with high accuracy if the reference points are densely detected. AcMu [6] exploits the static behaviors of mobile devices to update radio maps. By pinpointing mobile devices with a trajectory matching algorithm, the system employs them as mobile reference points to collect real-time RSS samples when they are static. CrowdAtlas [5] deals with the road variations in the map. It uses mobile navigation app to detect significant portions of GPS traces that do not conform to the existing map. When there is sufficient exceptional traces collected, map inference algorithms can automatically update the map. Existing works consider various dynamics in maps, while CrowdGIS copes with a novel kind of variation which updates stores in the map.

Relevant Map-based Systems. Location-based service is an extensively studied field in mobile computing. Most of the existing systems depend on an accuracy map to achieve high performance. OPS [32] localizes a distant object in the map through various sensors in smartphones. ParkSense [33] is a mobile sensing system to detect available parking spots in the map. Borealis [34] utilizes commodity smartphones to locate WiFi access points in real time. Moreover, FOLLOWME [35] is a navigation system to navigate the users to the same destination taken by an earlier traveler with an existing map. All these technologies provide various map-based services to users, while CrowdGIS underpins a primary support for them to maintain high performance in the long term.

VII. Conclusion

In this paper, we propose CrowdGIS, an automatic store self-updating system for digital maps that leverages street views and sensing data crowdsourced from mobile users. Compared with previous works, CrowdGIS represents the first attempt to automatically update stores in digital map. To realize this system, shooting positions of captured street views are first localized by a novel joint position estimation scheme. Second, store names are recognized through detecting either logo or text from street views. Third, we transfer the shooting position to the location of recognized stores in the map. Finally, CrowdGIS considers three updating categories to update them in the map based on the kernel density estimate model. We implement CrowdGIS and conduct extensive experiments in a real outdoor region for 1 month. The evaluation results demonstrate that CrowdGIS effectively accommodates store variations and maintains an up-to-date map.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China under Grant No. 61772446, and HK PolyU 4-ZZFF. The authors would also like to sincerely thank the editors and reviewers for their thoughtful, constructive suggestions and comments.

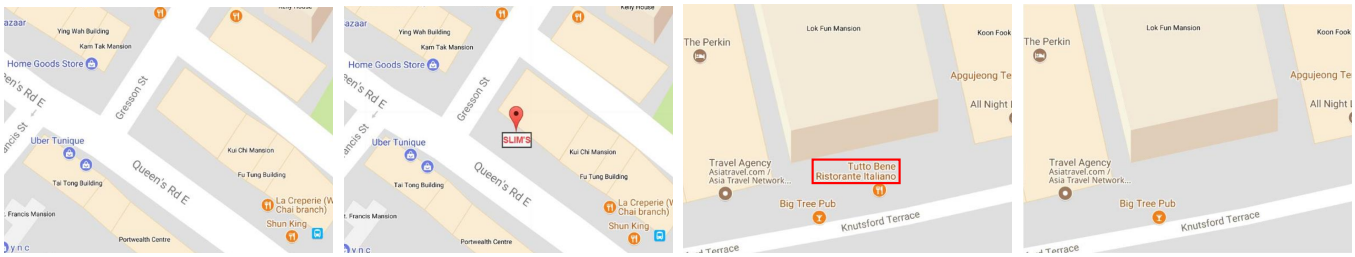
References

[1] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[2] J. Civera, M. Ciocarlie, A. Aydemir, K. Bekris, and S. Sarma, "Guest editorial special issue on cloud robotics and automation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 396–397, 2015.

[3] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, 2017.

[4] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, "Cloud-based collaborative 3d mapping in real-time with low-cost robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 423–431, 2015.



(a) Ground truth. (b) A store is added wrongly. (c) Ground truth. (d) A store is deleted wrongly.

Fig. 14: Failure cases: wrongly adding and deleting stores.

[5] Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, and Y. Zhu, "Crowdatlas: self-updating maps for cloud and personal use," in *Proc. of ACM Mobisys*, 2013.

[6] C. Wu, Z. Yang, C. Xiao, C. Yang, Y. Liu, and M. Liu, "Static power of mobile devices: Self-updating radio maps for wireless indoor localization," in *Proc. of IEEE INFOCOM*, 2015.

[7] S. Chen, M. Li, K. Ren, and C. Qiao, "Crowd map: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos," in *Proc. of IEEE ICDCS*, 2015.

[8] P. Sturm, "Pinhole camera model," in *Computer Vision*. Springer, 2014, pp. 610–613.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. of NIPS*, 2014.

[10] C. A. Vanegas, D. G. Aliaga, and B. Benes, "Automatic extraction of manhattan-world building masses from 3d laser range scans," *IEEE transactions on visualization and computer graphics*, vol. 18, no. 10, pp. 1627–1637, 2012.

[11] Z. I. Botev, J. F. Grotowski, D. P. Kroese *et al.*, "Kernel density estimation via diffusion," *The Annals of Statistics*, vol. 38, no. 5, pp. 2916–2957, 2010.

[12] K. Liu, Y. Li, F. He, J. Xu, and Z. Ding, "Effective map-matching on the most simplified road network," in *Proc. of ACM SIGSPATIAL GIS*, 2012.

[13] G. Pass, R. Zabih, and J. Miller, "Comparing images using color coherence vectors," in *Proc. of ACM MULTIMEDIA*, 1997.

[14] B. S. Manjunath and W.-Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.

[15] A. K. Jain, Y. Zhong, and S. Lakshmanan, "Object matching using deformable templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, pp. 267–278, 1996.

[16] J. Rife, S. Pullen, B. Pervan, and P. Enge, "Paired overbounding and application to gps augmentation," in *Proc. of IEEE PLANS*, 2004.

[17] L. Zhang, Y. Zhang, J. Tang, K. Lu, and Q. Tian, "Binary code ranking with weighted hamming distance," in *Proc. of IEEE CVPR*, 2013.

[18] Y. Yuan, B. Li, and M. Q.-H. Meng, "Wce abnormality detection based on saliency and adaptive locality-constrained linear coding," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 149–159, 2017.

[19] F. Yang, H. Lu, and Y.-W. Chen, "Human tracking by multiple kernel boosting with locality affinity constraints," in *Proc. of ACCV*. Springer, 2011.

[20] Y. Yuan and M. Q.-H. Meng, "Deep learning for polyp recognition in wireless capsule endoscopy images," *Medical Physics*, vol. 44, no. 4, pp. 1379–1389, 2017.

[21] Y. Yuan, X. Yao, J. Han, L. Guo, and M. Q.-H. Meng, "Discriminative joint-feature topic model with dual constraints for wce classification," *IEEE Transactions on Cybernetics*, 2017.

[22] C. Yao, X. Bai, and W. Liu, "A unified framework for multioriented text detection and recognition," *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4737–4749, 2014.

[23] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *Proc. of ACM SIGMOD*, 1996.

[24] J. Li, Z. Peng, S. Gao, B. Xiao, and H. Chan, "Smartphone-assisted energy efficient data communication for wearable devices," *Computer Communications*, vol. 105, pp. 33–43, 2017.

[25] S. Gao, Z. Peng, B. Xiao, Q. Xiao, and Y. Song, "Scop: Smartphone energy saving by merging push services in fog computing," in *Proc. of IEEE IWQoS*, 2017.

[26] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-markie: indoor pathway mapping made easy," in *Proc. of USENIX NSDI*, 2013.

[27] Y. Wen, X. Tian, X. Wang, and S. Lu, "Fundamental limits of rss fingerprinting based indoor localization," in *Proc. of IEEE INFOCOM*, 2015.

[28] F. Yang, Q. Zhai, G. Chen, A. C. Champion, J. Zhu, and D. Xuan, "Flash-loc: Flashing mobile phones for accurate indoor localization," in *Proc. of IEEE INFOCOM*, 2016.

[29] X. Liu, S. Zhang, B. Xiao, and K. Bu, "Flexible and time-efficient tag scanning with handheld readers," *IEEE Transactions on Mobile Computing*, vol. 15, no. 4, pp. 840–852, 2016.

[30] M. Alzantot and M. Youssef, "Crowdinside: automatic construction of indoor floorplans," in *Proc. of ACM SIGSPATIAL*, 2012.

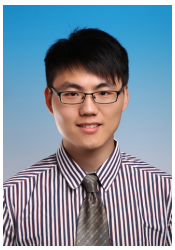
[31] J. Yin, Q. Yang, and L. M. Ni, "Learning adaptive temporal radio maps for signal-strength-based location estimation," *IEEE Transactions on Mobile Computing*, vol. 7, no. 7, pp. 869–883, 2008.

[32] J. G. Manweiler, P. Jain, and R. Roy Choudhury, "Satellites in our pockets: an object positioning system using smartphones," in *Proc. of ACM MobiSys*, 2012.

[33] S. Nawaz, C. Efstratiou, and C. Mascolo, "Parksense: A smartphone based sensing system for on-street parking," in *Proc. of ACM MobiCom*, 2013.

[34] Z. Zhang, X. Zhou, W. Zhang, Y. Zhang, G. Wang, B. Y. Zhao, and H. Zheng, "I am the antenna: accurate outdoor ap location using smartphones," in *Proc. of ACM MobiCom*, 2011.

[35] Y. Shu, K. G. Shin, T. He, and J. Chen, "Last-mile navigation using smartphones," in *Proc. of ACM MobiCom*, 2015.

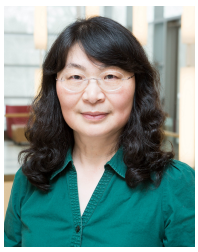


Zhe Peng is a Ph.D. candidate in Department of Computing, The Hong Kong Polytechnic University. He received his BSc degree in Communication Engineering from Northwestern Polytechnical University, and MSc degree in Electronic Engineering and Information Science from University of Science and Technology of China, in 2010 and 2013 respectively. His research interests include computer networks, mobile computing, data analysis, and machine learning.

Shang Gao received his B.S. degree from Hangzhou Dianzi University, China, in 2010, and his M.E. degree from Southeast University, China, in 2014. He is currently a PhD candidate with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests include information security, network security, and software-defined networks.

Bin Xiao is an associate professor at Department of Computing, the Hong Kong Polytechnic University, Hong Kong. Dr. Xiao received the B.Sc and M.Sc degrees in Electronics Engineering from Fudan University, China, and Ph.D. degree in computer science from University of Texas at Dallas, USA. After his Ph.D. graduation, he joined the Department of Computing of the Hong Kong Polytechnic University as an Assistant Professor. He has been the director of the Mobile Cloud Computing Lab in the department. His research interests include distributed wireless systems and security, mobile data analytics, software-defined networks (SDN) and network security. Dr. Xiao has published more than 100 technical papers in international top journals and conferences. He has been the symposium co-chair of IEEE ICC 2018 and Globecom 2017, and the general chair of IEEE SECON 2018. Currently, he is the associate editor of the Journal of Parallel and Distributed Computing (JPDC). He is the IEEE Senior member, a member of IEEE Communications Society and ACM member.

Songtao Guo received the BS, MS and the PhD degrees in computer software and theory from Chongqing University, China, in 1999, 2003 and 2008, respectively. He was a professor from 2011 to 2012 at Chongqing University. He was a senior research associate at City University of Hong Kong from 2010 to 2011, and a visiting scholar at Stone Brook University, New York, from May 2011 to May 2012. He is currently a professor at Southwest University, China. His research interests include wireless sensor networks, wireless ad hoc networks and parallel distributed computing. He has published more than 80 scientific papers in leading refereed journals and conferences. He has received many research grants as a principal investigator from the National Science Foundation of China and Chongqing and the Postdoctoral Science Foundation of China. He is a member of the IEEE.



Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a professor of computer engineering and computer science at Stony Brook University, New York, and the director in Communications and Devices Division, New York State Center of Excellence in Wireless and Information Technology (CEWIT). Her research interests include wireless networks, data center networks, optical networks, and highspeed networks. She has published more than 300 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She has served as an associate editor-in-chief and associated editor for the *IEEE Transactions on Computers* and an associate editor for the *IEEE Transactions on Parallel and Distributed Systems*. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is a fellow of the IEEE.