EyeLoc: Smartphone Vision Enabled Plug-n-play Indoor Localization in Large Shopping Malls

Zhichao Cao
School of Software
Tsinghua University
Beijing, China
caozc@tsinghua.edu.cn

Jialuo Du
School of Software
Tsinghua University
Beijing, China
dujl16@mails.tsinghua.edu.cn

Manni Liu
Computer Science and Engineering
Michigan State University
East Lansing, the U.S.
liumanni@msu.edu

Qing Zhou
School of Software
Tsinghua University
Beijing, China
zhouq16@mails.tsinghua.edu.cn

***Abstract*—Indoor localization is an emerging demand in many large shopping malls. Existing indoor localization systems, however, require exhausted system bootstrap and calibration phases. The huge sunk cost usually hinders practical deployment of the indoor localization systems in large shopping malls. In contrast, we observe that floor-plan images of large shopping malls, which highlight the positions of many shops, are widely available in Google Maps, Gaode Maps, Baidu Maps etc. According to several observed shops, people can localize themselves (*self-localization*). However, due to the requirements of geometric sense and space transformation, not all people get used to this way. In this paper, we propose EyeLoc, which uses smartphone vision to enable accurate self-localization on floor-plan images. EyeLoc addresses several challenges which include developing ubiquitous smartphone vision system, efficient vision clue extraction and robust measurement error mitigation. We implement EyeLoc in Android and evaluate its performance in emulated environment and four large shopping malls. The 90-percentile errors of localization and heading direction are 4m and 20° in the two large shopping malls.**

***Index Terms*—Indoor localization, smartphone vision system, inertial measurement, text detection/recognition.**

I. Introduction

Nowadays, the physical layout of many large shopping malls are becoming more and more complex [17]. As there are many location based activities (e.g., shopping, eating, watching movie) in large shopping malls, indoor localization is becoming an important service for people. Although outdoor localization (i.e., GPS) has been put in practice for many years, there is still no practical deployed indoor localization systems.

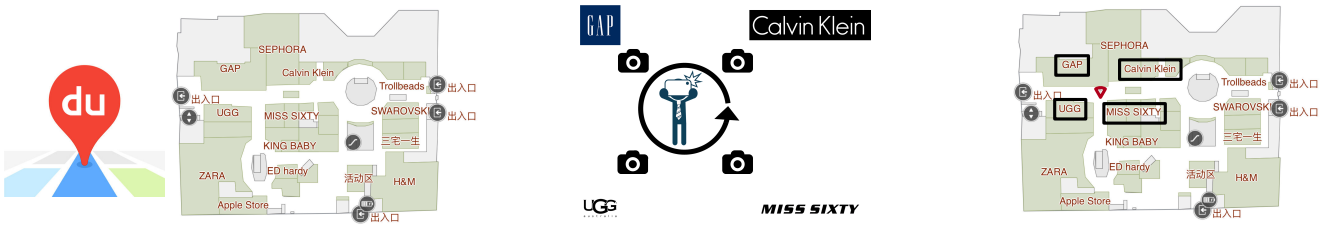
Many indoor localization systems rely on pre-collected information (e.g., Wi-Fi signals [9] [22] [6] [16] [21], lamp position [5] [24] [8], indoor environment images [10], magnetic field fingerprints [13] [19]), called *site survey*, to construct a localizable map. In large shopping malls, the site survey usually incurs extensive bootstrap overhead so that hinders existing approaches from being used. Even when site survey can be accomplished, the information usually needs to be timely updated and calibrated which further limits the applicability. Moreover, some indoor localization systems [16] require custom hardwares, which are not supported in commodity smartphones.

Therefore, the question is *can we setup a plug-and-play indoor localization system in large shopping malls nowadays*? We notice a possible way by leveraging the widely available floor-plan images, which can be obtained from indoor map providers (e.g., Google Maps, Gaode Maps, Baidu Maps, etc.). Those floor-plan images contain the positions of many shops, called POIs (Point of Interest). These POIs are often used as visual hints to help users manually localize themselves, called *self-localization*. Although there is no overhead of site survey, self-localization usually requires users have good ability of geometric sense and space transformation. To reduce users' mental work, we imagine that can users automatically obtain their positions on floor-plan image from their smartphones as traditional outdoor localization system such as Google Maps and Baidu Maps? In this sense, it is possible to achieve a plug-and-play indoor localization system by bridging this gap.

In this paper, we propose EyeLoc, a step towards plug-and-play indoor localization in large shopping malls. The key idea of EyeLoc is to imitate human self-localization with smartphone vision. After obtaining a floor-plan image, EyeLoc uses scene text detection/recognition techniques to extract a set of POIs. The recognized texts are used to identify different POIs. Moreover, the corresponding text bounding boxes provide the approximate positions of the POIs in the floor-plan coordinate system (called *floor-plan space*). Then, a user holds his/her smartphone and turns a 360° circle. The smartphone automatically shoots a series of images (called *view image*), which contain the surrounding POI signs. For those observed POIs, EyeLoc extracts their texts and geometric constraints in vision space, which are further used to match the user's position in floor-plan space.

EyeLoc addresses three challenges. First, there is a big difference between human vision system and smartphone vision system. We develop an accurate and ubiquitous monocular vision system which is available on most smartphones. Then, we construct the constant geometric constraints of 3 observed POIs to enable position matching between floor-plan space and vision space. Second, text detection and recognition are usually time-consuming. To reduce the processing time of POI extraction, an outlier image filtering method and a sparse image processing method are designed. Third, the measurement errors from motion sensors and floor-plan images may incur inaccurate position matching, for which we design an error-resilient method. We implement EyeLoc

978-1-7281-2376-9/19/$31.00 ©2019 IEEE



(a) Indoor Floor-plan Image (b) Circle Shoot (c) Location & Heading

Fig. 1: Illustration of an example of the EyeLoc innovation.

on Android smartphones and evaluate its performance in an office environment and two large shopping malls (7,500m^2 and 10,000m^2). The evaluation results show that the 90-percentile error of localization and heading direction can achieve 4m and 20°. The contributions of this paper are as follows.

- We propose EyeLoc, a smartphone vision enabled plug-and-play indoor localization in large shopping malls. No site survey or periodical system calibration is required.
- We develop a ubiquitous smartphone vision system and corresponding geometric localization model. To guarantee localization accuracy and processing efficiency, we propose the countermeasures to address several practical challenges.
- We implement EyeLoc on Android smartphones and evaluate its performance in an office environment and two large shopping malls. The evaluation results show that EyeLoc is effective on both localization accuracy and processing efficiency.

The rest of this paper is organized as follows. Section II shows the overview of EyeLoc. Section III illustrates the detailed design of EyeLoc. Section IV and Section V show the details of EyeLoc implementation and evaluation, respectively. Section VI introduces the related work. Finally, we conclude our work in Section VII.

II. EyeLoc Overview

GPS is a plug-and-play localization system in outdoor spaces and has been widely used by smartphone applications. In large shopping malls, EyeLoc has two goals as GPS does.

- *plug-and-play*. EyeLoc should not assume any extra bootstrap cost (e.g., site survey, system calibration) in large shopping malls. Moreover, EyeLoc should not require users to own any prior knowledge and follow complex smartphone operations.
- *efficient and robust*. Facing computation intensive image processing and various measurement errors from motion sensors and floor-plan images, EyeLoc should be able to accurately localize a user with short processing time.

To meet the first goal, EyeLoc is inspired by two observations. First, the indoor floor-plan images of shopping malls (e.g., shown in Figure 1(a)) can be easily fetched from indoor map providers through Android and IOS APIs. The other observation is that people usually turn around to observe the surrounding POIs and localize themselves. After fetching the floor-plan images, Eyeloc enables the smartphone to imitate human self-localization with on-board motion sensors and technologies of text detection and recognition. Bootstrap cost and user training are no longer in need.

Fig. 1 provides an example of how EyeLoc works. Alice is in a large shopping mall and she wants to go to H&M. She loses her direction and does not know where she is. As Alice opens Eyeloc, the corresponding floor-plan image is automatically fetched. Then she holds her smartphone and turns a 360° circle. EyeLoc takes a set of view images during Alice's turning. This operation is called *circle shoot*. From view images, Eyeloc exploits the motion sensors and techniques of text detection and recognition to extract geometric information of several observed POIs (e.g., GAP, UGG, MISS SIXTY, Calvin Klein). Meanwhile, EyeLoc uses techniques of text detection and recognition to find the positions of corresponding POIs on the floor-plan image. Finally, Eyeloc projects Alice's position and heading direction on floor-plan image as shown in Fig. 1(c).

As Fig. 1 illustrates, EyeLoc depends on techniques of text detection and recognition to extract geometric information of the observed POIs from a set of view images. Text detection and recognition from color images have been widely studied in the past decade, especially with deep learning models like convolutional neural networks. A few open-source models (e.g., OpenCV [1] , Tesseract [15]) are also available on smartphones. On the other hand, smartphones can utilize the cloud service from companies like Google, Baidu, etc. However, none of the two approaches can achieve real-time execution due to computation overhead on images or extra network delay. This contradiction obliges us to design an efficient method to accurately extract the geometric information of the observed POIs, but shorten the processing latency. We have two intuitions for the method design. First, since the extracted POIs with error geometric information are useless even harmful for localization, we should not deal with those view images of low quality. Second, we observe that the view images are usually redundant for extracting the geometric information of an observed POI. Hopefully, we can only select a subset of those view images which contains equivalent geometric information of the observed shops as the whole set does for further processing.

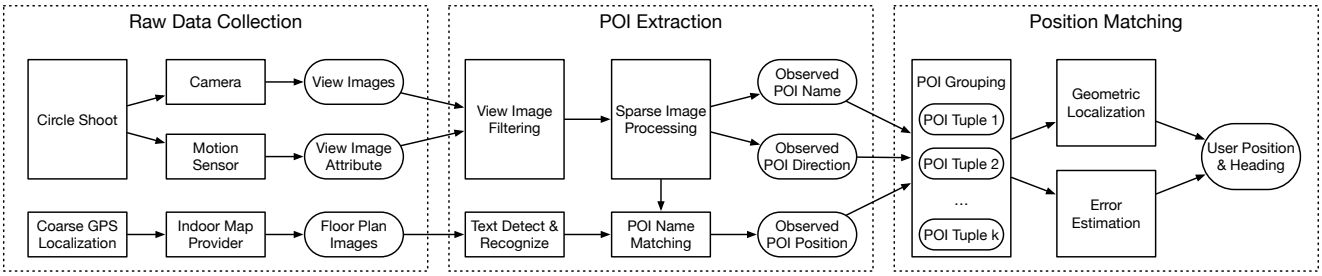


Fig. 2: Illustration of the system architecture of EyeLoc.

On the other hand, various measurement errors of the extracted information are invertible and may lead to inaccurate localization. For example, as shown in Fig. 1(c), the text bounding boxes of the four observed shops may be not exactly align to that of the corresponding shop signs appeared in vision space. To mitigate potential errors and achieve robust localization, our observation is that the spatial POI distribution is usually dense in large shopping malls, which means multiple POIs are available. Hopefully, we can use the the redundant information to refine the estimated user's position.

In comparison with human binocular vision system, EyeLoc develops a monocular vision system, which is accurate and ubiquitous for smartphones. EyeLoc enables user's position matching between vision space and floor-plan space with constant geometric constraints of 3 observed POIs. The system architecture of Eyeloc is shown in Figure 2, including three parts as follows.

Raw Data Collection. The first part is to fetch floor-plan images from indoor map providers and collect raw information of view images from circle shoot. According to the coarse GPS localization, Eyeloc queries indoor map providers to obtain floor-plane images. During the circle shoot, Eyeloc uses camera and motion sensors (e.g., compass, gyroscope, accelerometer) to continuously capture view images and corresponding motion attributes (e.g., camera facing direction, angle velocity).

POI Extraction. Taking the information of view images and floor-plan images as input, the second part extracts geometric information of several observed POIs in both vision space and floor-plan space. Because text detection and recognition are time-consuming, we need to extract enough geometric information while keeping the number of processed image as small as possible. Eyeloc filters out some view images which are blurred or have error motion attributes. Then EyeLoc develops a sparse image processing method to extract geometric information of all observed POIs from the rest of view images and keeps the number of processed image small. After extracting all POIs on floor-plan image, EyeLoc obtains the positions of the observed POIs in floor-plan space by matching their names.

Position Matching. With the geometric information of the observed POIs, EyeLoc now projects the user's position and heading direction to floor-plan space. The redundancy of the observed POIs is explored to mitigate unavoidable errors of geometric information in vision space and positions in floor-plan space. The observed POIs are grouped into tuples. Each POI tuple can be used to calculate the user's position and heading direction with geometric constraints. The localization error of different tuples are diverse with the same measurement errors. EyeLoc combines several inferred positions and the corresponding errors to vote the final user's position and heading direction.

III. EyeLoc Design

To relieve human from self-localization, EyeLoc achieves plug-and-play localization in large shopping malls. Due to the huge difference between human vision system and smartphone vision system, we first establish the smartphone vision system and illustrate the fundamental geometric localization model in EyeLoc. Then we show the design details of the three function components of EyeLoc as shown in Figure 2.

A. Smartphone Vision System

We intend to define a ubiquitous smartphone vision system to fetch the geometric relationship between the smartphone and an observed POI. Human eyes form a binocular vision system, which enables humans to estimate the distance of an observed POI and the direction of the sightline between the POI and human eyes. However, nowadays only a few smartphones (e.g., iPhone X, Huawei P20 Pro) have dual or triple cameras and parameters of camera calibration are not explicitly known. Moreover, humans have practiced a lot since they were children. So the question is *can we estimate the distance and the direction as geometric descriptors of an observed object through the monocular view images of circle shoot?*

Fig. 3: An example of the sightline change during circle shoot in physical space.

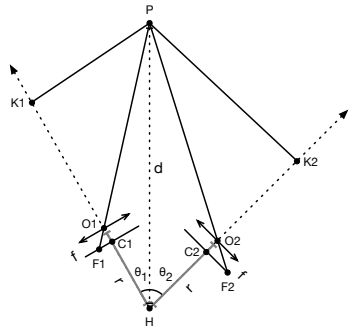
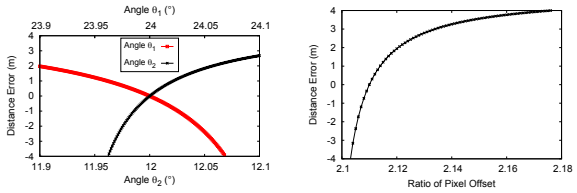


Fig. 4: Distance measurement with circle shoot.



(a) Influence of θ_1 and θ_2 (b) Influence of k

Fig. 5: Illustration of distance error in terms of the error of θ_2 and k. (a) and (b) show the distance error under different θ_2 and k when other parameters are fixed.

To enable distance estimation with monocular vision, one way is to exploit the camera motion of circle shoot to imitate a binocular vision system. As shown in Fig. 4, the distance between the POI P and the user H is indicated as d. The distance between H and the optical center of smartphone camera lens O_1 is r. C_1 is the center of image plane while F is the position of P on the image. H_1, C_1 and O_1 are approximately kept on the same line all the time during circle shoot. The focal length of the smartphone camera lens f is unknown for most smartphones. θ_1 indicates the intersection angle between line HO_1 and the sightline HP. Since $\triangle F_1O_1C_1$ is similar to $\triangle PO_1K_1$, we have the follow equation:

$$\frac{F_1C_1}{d\sin\theta_1} = \frac{f}{d\cos\theta_1 - r} \tag{1}$$

where F_1C_1 indicates the pixel offset between F_1 and $C1$. Combining the same equation under another angle θ_2 ($\theta_1 \neq \theta_2$), we can derive d as follow:

$$d = r\frac{\sin\theta_1 - k\sin\theta_2}{\cos\theta_2\sin\theta_1 - k\cos\theta_1\sin\theta_2} \tag{2}$$

where k equals to the ratio between F_1C_1 and F_2C_2. If θ_1, θ_2, k and r are known, the distance d can be calculated. As the direction of HP, HO_1 and HO_2 can be obtained by motion sensors, θ_1 and θ_2 can be calculated. For an observed POI, we recognize the center of its text bounding box as F_1 and F_2 on a view image so that F_1C_1, F_2C_2 and k can be calculated. r can be roughly estimated according to human arm length. In this way, EyeLoc can estimate the distance of a POI in large shopping malls without any prior knowledge of camera parameters.

Since the errors of θ and k estimation are inevitable, we conduct error analysis. We assume r is 0.5m. Given θ_1, θ_2 and k are 24°, 12° and 2.11, d will be 5.48m. We change one parameter (e.g., θ_1, θ_2, k) to calculate the distance error when other parameters are fixed. The results are shown in Fig. 5. Surprisingly, given the distance as 5.48m, the distance error is huge when θ_1, θ_2 and k have a small bias. The distance error is getting to 1.97m, when θ_1 decreases from 24° to 23.9°. Similarly, when θ_2 increases from 12° to 12.1°, the distance error increases from 0m to 2.68m. 0.1° error is common for the facing direction measurement with motion sensors. The same trend happens on k. When k increases from 2.11 to 2.16, the distance error increases from 0m to 3.77m. Due to the limitation of image resolution, given F_1C_1 is as large as 1000 pixels, 0.05 error of k means about no more than 50 pixels error of F_2C_2 which is hard to achieve due to the relatively large estimation error of text bounding box. When θ_1 increases or θ_2 and k decreases a little, the situation is getting even worse. Hence, due to the limitation of motion sensor precision and image resolution, the potential huge error makes the distance estimation unavailable in practice. Overall, in smartphone vision system, for a POI, we only use the direction of its EyeLoc sightline as the geometric descriptor for later localization.

Algorithm 1 Geometric Constraints Extraction Algorithm

Input: 3 POIs sorted according to their appearance order; the directions of the corresponding EyeLoc sightline δ_1, δ_2, δ_3 in vision space.

Output: rotation directions d_{12}, d_{23}, d_{31}, intersection angles θ_{12}, θ_{23} and θ_{31}.

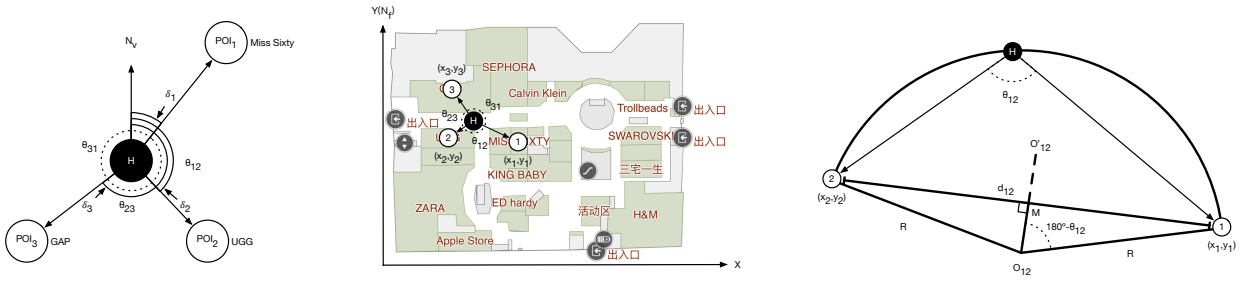
1: vector of POI_1 EyeLoc sightline $v_1 = (\sin\delta_1, \cos\delta_1)$.
2: vector of POI_2 EyeLoc sightline $v_2 = (\sin\delta_2, \cos\delta_2)$.
3: vector of POI_3 EyeLoc sightline $v_3 = (\sin\delta_3, \cos\delta_3)$.
4: $d_{12} = v_1 \times v_2$, $d_{23} = v_2 \times v_3$ and $d_{31} = v_3 \times v_1$.
5: $\theta_{12} = (\delta_2 - \delta_1) \mod 360°$; $\theta_{23} = (\delta_3 - \delta_2) \mod 360°$; $\theta_{31} = (\delta_1 - \delta_3) \mod 360°$

B. Geometric Localization Model

After we obtain the direction of the sightlines of several POIs, the next question is *how to construct constant geometric constraints, then figure out the user's position and heading direction on floor-plan image.*

1) Constant Geometric Constraints: Taking Fig. 6 as an example, H is the user's position. Fig. 6(a) represents the vision space. The user observes 3 POIs (e.g., POI_1 Miss Sixty, POI_2 UGG and POI_3 GAP) in their appearance oder. N_v indicates the north direction in vision space. Fig. 6(b) represents the floor-plan space. Circle 1, 2 and 3 in Fig. 6(b) represent the corresponding center of text bounding boxes of the 3 observed POIs. Given the coordinate system X-Y of the floor-plan image, (x_1, y_1), (x_2, y_2) and (x_3, y_3) are the corresponding coordinate of 1, 2 and 3. N_f is the north direction in floor-plan space which aligns with Y axis. In vision space, the directions of EyeLoc sightline δ_1, δ_2 and δ_3 can be estimated. However, since N_v and N_f may be not aligned with each other, we cannot directly determine the coordinate of H with these directions in floor-plan space.

Two constant geometric constraints between the vision space and the floor-plan space are important for EyeLoc. The first one is that the rotation direction of the circle shoot is constant. The 3 POIs will appear in the same order along the rotation direction in vision space and floor-plan space (e.g., $\text{POI}_1 \rightarrow \text{POI}_2 \rightarrow \text{POI}_3$ in Fig. 6(a) and $1 \rightarrow 2 \rightarrow 3$ Fig. 6(b)). We use d_{12}, d_{23} and d_{31} to indicate rotation directions between each pair of adjacent POIs. The second constant geometric constraint is, $\angle\text{POI}_1H\text{POI}_2$, $\angle\text{POI}_2H\text{POI}_3$ and $\angle\text{POI}_3H\text{POI}_1$ in Fig. 6(a) are equal to $\angle 1H2$, $\angle 2H3$ and $\angle 3H1$ in Fig. 6(b) respectively. To simplify our notations, the 3 intersection angles are indicated as θ_{12}, θ_{23} and θ_{31} in both spaces. The rotation directions and the intersection angles between any two POIs serve as two constant geometric constraints between the vision space and the



(a) Direction in vision space (b) Direction in floor–plan space (c) Possible location on an arc

Fig. 6: Illustration of the model used to localize a user's position on floor-plan image with 3 observed POIs. (a) and (b) exhibit a constant geometric constraint in both vision space and floor-plan space. (c) shows the model to calculate the user's location with the extracted geometric information.

Algorithm 2 Arc Calculation Algorithm

Input: 2 POIs, POI_1 and POI_2, sorted according to their appearance order; the rotation direction d_{12}; the angle θ_{12} between the directions of the corresponding EyeLoc sightlines in vision space; the corresponding coordinates (x_1, y_1) and (x_2, y_2) in floor-plan space.

Output: The coordinate of circle center (x_o, y_o); the length of circle radius R.

pixel distance and slope of the chord $\vec{12}$ as $d_{12} = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$ and $k = \frac{x_1-x_2}{y_1-y_2}$.
if θ_{12} equals to 180° **then**
  H is on the segment between POI_1 and POI_2. (x_o, y_o) and R are set as NULL.
else if $\theta_{12} > 180^\circ$ **then**
  $\theta_{12} = 360^\circ - \theta_{12}$.
else if θ_{12} equals to 90°. **then**
  $(x_o, y_o) = (\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$;$R = d_{12}/2$
else
  $R = \frac{d_{12}}{2\sin\theta_{12}}$; two possible coordinates (x_{o1}, y_{o1}) and (x_{o2}, y_{o2}) of circle center are calculated by Equ. 6 and Equ. 6.
  set (x_o, y_o) as (x_{o1}, y_{o1})
  calculate the rotation direction $d_{o12} = \text{vector}(x_o - x_1, y_o - y_1) \times \text{vector}(x_o - x_2, y_o - y_2)$
  if $d_{12} \cdot d_{o12} > 0 \oplus \theta_{12} < 90^\circ$. **then**
    set (x_o, y_o) as (x_{o2}, y_{o2}).
  end if
end if

floor-plan space. Alg. 1 exhibits the details to determine the d_{12}, d_{23}, d_{31}, θ_{12}, θ_{23} and θ_{31} given 3 POIs and their corresponding directions of EyeLoc sightlines.

2) Localization Model: Given three POI coordinates $((x_1, y_1)$, (x_2, y_2), $(x_3, y_3))$, rotation directions (d_{12}, d_{23}, d_{31}) and intersection angles $(\theta_{12}, \theta_{23}, \theta_{31})$, we derive a method to calculate the coordinate (x_H, y_H) of the user's position H in the floor-plan space. As shown in Fig. 6(c), we have the coordinates of two POIs (e.g., 1, 2), the rotation direction d_{12} and the intersection angle θ_{12}. If θ_{12} is 180°, H is on the segment between 1 and 2. Otherwise, the possible position of H is on an arc which takes the segment $\vec{12}$ as the chord and θ_{12} as the inscribed angle. The pixel distance between 1 and 2 is l_{12} which equals to $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$. M is the middle point of the chord $\vec{12}$ and its coordinate (x_M, y_M) equals to $(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$. O_{12} is the center of the circle and R is the length of its radius. We use (x_o, y_o) to indicate the coordinate of O_{12}. If θ_{12} is 90°, O_{12} and M will share the same coordinate. Otherwise, since 1 and 2 are on the circle and the chord $\vec{12}$ is perpendicular to MO_{12}, we have following equation:

$$\frac{x_1 - x_2}{y_1 - y_2} = -\frac{y_o - y_M}{x_o - x_M} = k \tag{3}$$

where k is the slope of the chord $\vec{12}$. Moreover, the central angle $\angle 1O_{12}2$ is twice the corresponding inscribed angle which equals to $180^\circ - \theta_{12}$ and $\angle 1O_{12}M$ is half the central angle $\angle 1O_{12}2$. Hence, $\angle 1O_{12}M = 180^\circ - \theta_{12}$ and $R = \frac{d_{12}}{2\sin\theta_{12}}$. As for the length of $O_{12}M$, we have the follow equation:

$$\sqrt{(x_o - x_M)^2 + (y_o - y_M)^2} = -\frac{l_{12}}{2\tan\theta_{12}} \tag{4}$$

Combining Equ. 3 and Equ. 4, we can obtain (x_o, y_o) as follows:

$$x_o = x_M \pm \frac{l_{12}}{2\tan\theta_{12}\sqrt{1+k^2}} \tag{5}$$

$$y_o = y_M \mp \frac{kl_{12}}{2\tan\theta_{12}\sqrt{1+k^2}} \tag{6}$$

Besides O_{12}, we obtain another false center of circle O'_{12} which is symmetric with O_{12} by taking $\vec{12}$ as mirror. To filter out the outlier O'_{12}, we further exploit the information of the rotation direction d_{12} and acute/obtuse angle θ_{12}. If θ_{12} is acute angle, O_{12} is on the same side with H regarding segment $\vec{12}$. Otherwise, O_{12} is on the opposite side with H. In this way, we can identify the unique coordinate of O_{12}. Alg. 2 summarizes the detailed calculation of O_{12} and R. Now, we know H is on an arc determined by O_{12} and R. Similarly, we can calculate another arc where H is on with POI_2, POI_3 and θ_{23}. We further calculate the intersections of these two arcs. One intersection is POI_2, the other is the position of H.

In the localization model, we should notice an unexpected situation which may incur localization bias. The situation is that when H, POI_1, POI_2 and POI_3 are on the same circle, we cannot localize H through the geometric constraints of the 3 POIs. This situation rarely happens in practice as shown in Section V. Moreover, it is possible that we can observe more than 3 POIs at large shopping malls. If the situation has happened, EyeLoc will popup a message to remind the user that he/she needs to walk several steps and relocalize himself/herself.

When the coordinate of H is known, we can calculate the directions of $H\text{POI}_1$, $H\text{POI}_2$ and $H\text{POI}_3$ in floor-plan space. Then, with δ_1, δ_2 and δ_3, we can calculate the angle offset Δ_N between the vision north N_v and floor-plan north N_f. Given any user's heading direction (e.g., camera facing direction) in vision space, we can infer his/her heading direction in floor-plan space. Overall, EyeLoc can calculate a user's position and heading direction by observing no less than 3 POIs. In the next 3 subsections, we will address several challenges in practice.

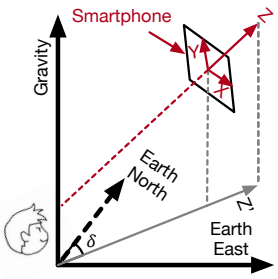


Fig. 7: Illustration of the camera facing direction δ measurement.

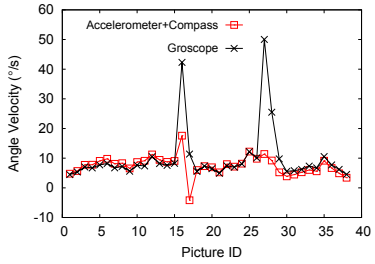


Fig. 8: The angle velocity measured by different combination of motion sensors.

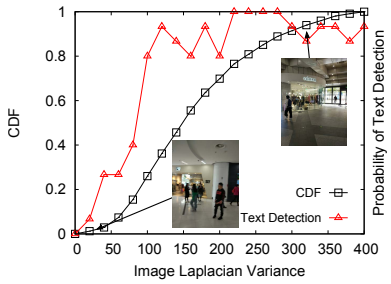


Fig. 9: The influence of image blur on the accuracy of text detection.

C. Raw Data Collection

During circle shoot, EyeLoc periodically takes view images with camera and records the readings of motion sensors to infer the camera facing direction of each view image in vision space. Moreover, EyeLoc fetches the floor-plan images from indoor map provider.

1) View Image: We have two system parameters for view image shooting. One is the resolution of view image, indicated as I_r. The higher the I_r is, the text of more POIs can be accurately detected and recognized. However, the processing time also increases when I_r becomes high. Hence, EyeLoc fixes I_r as 1536p during circle shoot but adaptively chooses a small resolution I_p for later POI text detection and recognition. The other parameter is the shooting frequency f_s. The interval between two adjacent view images is $1/f_s$. The high f_s ensures all surrounding POIs can be recorded when the rotation speed of a user is fast. Too many redundant view images also have negative influence on processing time. EyeLoc selects a relatively high f_s to guarantee the reliability and further improves the processing efficiency through view image filtering and selection.

2) Motions Sensor Readings: In Section III-B, we have mentioned that the EyeLoc direction of each view image is measured through the estimation of camera facing direction. According to the common gesture of circle shoot shown in Fig. 3(a), as Fig. 7 shown, δ is the angle between earth north N_e (i.e., N_v in EyeLoc vision system) and the projected direction Z' of smartphone Z axis. We use the motion sensors (e.g., accelerometer, gyroscope and compass) to capture the camera facing direction. EyeLoc continuously samples the readings of the motion sensors. Basically, the acceleration along 3 smartphone axes (e.g., X, Y and Z) can determine the direction of gravity. Compass can further determine the direction of N_e in smartphone coordinate system. As a result, the direction of Z can be calculated in earth coordinate system. Then, the direction of Z' and δ are calculated correspondingly. Furthermore, EyeLoc adopts several methods [23] [11] to calibrate the camera facing direction of each view image by removing potential magnetic interference and bursty noise.

3) Floor-plan Image: . By utilizing the API provided by indoor map provider, given the coarse GPS readings in a shopping mall, EyeLoc can fetch the floor-plan images of all floors in the shopping mall. Each floor-plan image contains the skeleton and name of all POIs on that floor.

Overall, the raw data collection module outputs a series of view images, corresponding EyeLoc sightline directions and floor-plan images. However, in raw data, the redundancy and measurement error incur computation inefficiency and localization error. Next, we introduce the methods to improve the efficiency and robustness.

D. POI Extraction

EyeLoc fetches the names and positions of all POIs in a shopping mall from floor-plan images. The problem is to efficiently extract all available POIs and corresponding geometric information from a set of view images for later position matching.

1) View Image Outlier Filtering: If a view image is blurred, we cannot detect any text at all. We treat the blurred view images as outlier that should be filtered out. EyeLoc adopts Laplacian-based operator, which is a widely used function for focus measure, to define the degree of image blur. We randomly selected 1692 view images from the whole data set shot in two large shopping malls (Section V). We guarantee there are at least one POI sign in each of these view images. In Fig. 9, the black curve shows the Laplacian variance values of these images are distributed from 0 to 400. The larger the variance is, the less the image blur is, as shown the comparison between the example view images with variance in the range [0,20] and [300,320]. In a view image, if the length of any recognized text string is more than 2, it is text detectable. We further select 15 view images from each level of image blur to evaluate the the probability of text detection under different level of image blur. The red curve shows that the probability of text detection is higher than 80% when the Laplacian variance is larger than 80. Hopefully, we should filter out those view images whose Laplacian variance is less than 80 because it is hard to detect any text clues from it. Thus, we define a threshold Δ_{Lap} (e.g., approximate 80) to determine whether a view image is blurred or not.

Moreover, the smartphone vibration around Y axis and Z axis can influence the position of a POI on view images. As a result, the estimation error of EyeLoc sightline may increase. The gyroscope outputs the angle velocity around 3 axes as ω_x, ω_y and ω_z, then the angle velocity is $\omega = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$. On the other hand, we can also calculate the angle velocity ω' given the Z' direction of two adjacent view images and the corresponding time interval. We conduct a circle shoot by setting f_s as 2Hz. During circle shoot, we manually vibrate the smartphone as a common user does when the picture ID is from 15 to 18 and from 26 to 30. As shown in Fig. 8, the angle velocity difference between ω and ω' is close to zero as usual. However, the smartphone vibration will obviously increase the difference. This observation indicates different motion sensors have different sensitivity for the vibration. Hence, EyeLoc sets a threshold Δ_ω and filters those view images when the angle velocity difference is larger than Δ_ω.

2) Text Filtering and Matching: In large shopping malls, text may appear at anywhere. It is possible to detect multiple text strings from a view image, especially the name of a POI may appear at multiple places. EyeLoc filters out the irrelevant and duplicate text bounding boxes through following steps. First, given the minimum and maximum length of POI names extracted from floor-plan images, EyeLoc filters out the illegal text strings. Second, we group the rest of text strings. Two text strings belong to the same group when the difference between them is smaller than a threshold Δ_t. The difference between two text string is defined as the ratio between their Levenshtein Distanceand the maximum string length. Given the

Fig. 10: Feature point matching in two different view images of the same POI.

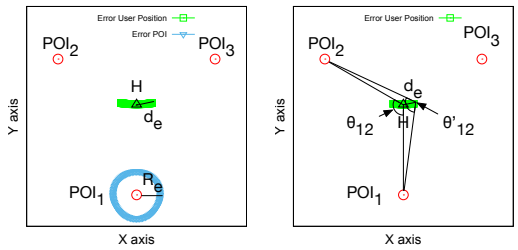


Fig. 11: The localization sensitivity regarding to (a) POI error and (b) direction error.

list of POI names extracted from floor-plan images, EyeLoc further removes those invalid groups whose text strings are not on the list (i.e., the similarity is smaller than Δ_t in comparison with any POI name). Finally, in each valid group, EyeLoc combines the coordinates of all text bounding boxes to calculate the average value as the unique text bounding box position of the observed POI on the view image. In this way, EyeLoc identifies the available POIs and corresponding position of text bounding boxes on a view image.

3) Sparse Image Processing: After filtering the outliers of view images, for all observed POI, we need to exactly find the view images (e.g, Fig. 3(c)) where the corresponding text bounding boxes appear in the middle of. The intuitive approach is to process all view images, but incurs heavy networking and computation burden as the sampling frequency f_s is set high. Even worse, the desired view image may not be captured or blurred. Instead of processing every view image to extract the geometric information of all potential POIs, EyeLoc develops a sparse image processing approach to achieve the same goal. The key idea is after the position of a text bounding box is known from a view image (e.g., Fig. 3(b)), we can enable EyeLoc sightline estimation of a POI with one more view image which contains the same POI (e.g., Fig. 3(d)) by feature point matching. As shown in Fig. 10, given two view images I_1 and I_2, the text bounding box of I_1 is extracted and it is d_1 pixel from the middle line. Then, in I_2, we use ORB algorithm to extract the same feature points which falls into the text bounding box of I_1. Given a feature point, its coordinates on I_1 and I_2 are (x_1, y_1) and (x_2, y_2). The pixel distance of the feature point is $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$. The average pixel distance of all feature points is indicated as l_f. Due to the approximate constant ratio between pixel distance and central angle, given their direction as δ_1 and δ_2, we can calculate the direction δ of the POI EyeLoc sightline as following:

$$\delta = \delta_1 + \frac{d_1}{l_f}(\delta_2 - \delta_1) \tag{7}$$

When we recognize the text bounding box of a POI from a view image, we use its adjacent images which probably contain the same POI to calculate the direction of POI EyeLoc sightline with Equ. 7.

To extract the geometric information of all observed POIs, the problem becomes to quickly find a view image of each observed POI from all view images. Given n view images $\{I_1, I_2, ..., I_n\}$, we set a step length Δ_s and view images $\{I_{\Delta_s}, I_{2\Delta_s}, ..., I_{k\Delta_s}\}$ $(k = \lfloor \frac{n}{\Delta_s} \rfloor)$ are selected for processing. Hopefully, if the minimum number of view images of a POI is larger than Δ_s, EyeLoc cannot miss the view image of any observed POI. However, due to the possible fail of text recognition, we may miss some POIs so that no more than three POIs are extracted. In this case, EyeLoc will exponentially reduce Δ_s and reprocess the new view images until at least 3 POIs are extracted or all view images are processed. Overall, we can fetch enough available POIs and corresponding directions of EyeLoc sightline as soon as possible for later location matching. Next, we remove the potential estimation errors to achieve accurate location matching.

E. Position Matching

According to the localization model in Section III-B, we can localize a user's position with three observed POIs, called *localization tuple*. Fig. 11 shows the measured POI coordinates of a localization tuple (e.g., POI_1, POI_2, POI_3) and the calculated user's position H. The corresponding measured intersection angle θ_{12}, θ_{23} and θ_{31} are 120°. The POI text bounding boxes in floor-plan image may not exactly align with that in physical space. Due to the POI coordinate errors, we assume the true POI positions may appear on a circle around it and the radius is R_e. As shown in Fig. 11(a), given the POI_1 uncertainty R_e as 3 pixels, we fix other measured results, the possible true positions shown as the green marks are calculated. Moreover, due to the possible errors from motion sensor and image processing, θ_{12} may inaccurately measured in comparison with the true intersection angle θ'_{12} as shown in Fig. 11(b). The same situation may happen for θ_{23} and θ_{31}. We assume the error of θ_{12}, θ_{23} and θ_{31} is in the range of $[-\Delta_\theta, \Delta_\theta]$. For θ_{12}, Fig. 11(b) shows the possible true positions shown as the green marks when Δ_θ is 10 °. Regarding to the errors of POI_1 and θ_{12}, the maximum localization errors are indicated as d_e. The ratio between d_e and R_e or Δ_θ is further defined as the error sensitivity of a POI or an intersection angle. Given a localization tuple u, we define its *localization error sensitivity* $les(u)$ as the sum of the error sensitivity of all three POIs and three intersection angles.

When k $(k \geq 3)$ POIs are extracted, we have total $m = k(k-1)(k-2)/6$ localization tuples indicated as $\{u_1, u_2, ..., u_m\}$. For the i^{th} tuple u_i, its localization result and error sensitivity are indicated as h_i and $les(u_i)$. The larger the $les(u_i)$ is, the more accurate the h_i is. Hence, EyeLoc sets the weight w_i of localization result h_i as $1/les(u_i)$. Then, the final match location h are calculated as follows:

$$h = \frac{\sum_{i=1}^{m} w_i h_i}{\sum_{i=1}^{m} w_i} \tag{8}$$

With h and k extracted POIs, EyeLoc can further calculate user's heading direction according to the method in Section III-B. Overall, EyeLoc outputs the user's location and real-time heading directions in floor-plan space.

IV. Implementation

We implement EyeLoc as a mobile application in Android 7.0. Fig. 12 demonstrates the user interface (UI) of EyeLoc application when we conduct experiments in the office environment (Section V). As shown in Fig. 12(a), after a user opens EyeLoc application, the surrounding view appears on the smartphone screen. Then, the user clicks "AUTOTAKEPHOTO" button to trigger circle shoot. After the user finishes circle shoot, he/she clicks "STOPTAKEPHOT" button. Later on, EyeLoc exhibits the user's position (e.g., black spot) and heading direction (e.g., orange arrow) on floor-plan image as shown in Fig. 12(b). We discuss several system details and settings as follows.

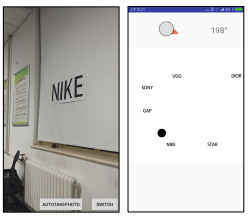
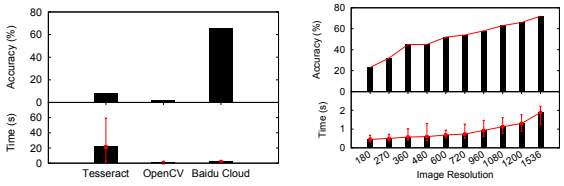


Fig. 12: The EyeLoc UI.



(a) Different text recognition approaches

(b) Different image resolution choices

Fig. 13: The performance under different text recognition approaches and different image resolution choices.

A. Scene Text Detection and Recognition

Scene text detection and recognition techniques serve as a fundamental role in EyeLoc. We compare several existing techniques which can work on Android smartphone in terms of recognition accuracy and processing time. Here, we adopt the same method in Section III-D to judge the similarity between two text string. Δ_t is set as 50%. We randomly select 100 images which are shot by smartphone in two large shopping malls. Some of them are shot at daytime and the others are shot at night. Each image contains one shop sign which is manually labeled as the ground truth.

1) Local processing v.s. Cloud processing: According to the different processing platform, there are two kinds of approaches for text detection and recognition. One is to locally process image on smartphone. The other is on cloud. The approaches of local processing include OpenCV [1] and Tesseract [15]. We choose Baidu Cloud as a typical cloud processing approach. We use LTE network, which is available in most shopping malls nowadays, to connect the smartphone with cloud server. Given the data set of 100 images, the recognition accuracy and processing time are shown in Fig. 13(a). We can see that the text recognition accuracy of Baidu Cloud is 66% which is much higher than 12% of Tesseract and 2% of OpenCV. The text recognition accuracy of Tesseract and OpenCV is surprisingly low since the text classifiers and extreme region extraction are hard to adapt the complex lighting condition and text format of the POI signs. The average processing time of Baidu Cloud is 2s which is a little higher than that of OpenCV, but much smaller than Tesseract. Due to the superior recognition accuracy and low processing time, we choose cloud processing instead of local processing.

2) The influence of image resolution: We further explore the performance of Baidu Cloud by using different image resolution. We vary the resolution of 100 images from 180p to 1538p. The performance is shown in Fig. 13(b). We can see that both text recognition accuracy and processing time increase with the increasing of image resolution. When the image resolution is 720p, the text recognition accuracy and average processing time are 54% and 0.74s. In comparison, the text recognition accuracy and average processing time increase to 72% and 1.89s when the image resolution increases to 1536p. EyeLoc choose I_p to keep the text recognition accuracy is higher than 60%. Meanwhile, EyeLoc minimizes the expected time for successfully recognizing a POI which is the ratio between the average processing time and the recognition accuracy. Hence, we set I_p as 1080p. For outlier view image filtering, according to the observation of Fig. 9 and 8, we set Δ_{Lap} and Δ_ω as 80 and $10^\circ/s$.

B. Circle Shoot Operation

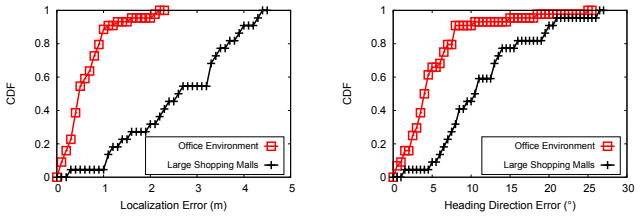
We set f_s as 2Hz, namely EyeLoc shoots 2 view images per second. Moreover, we set Δ_s as 3. Considering the observed 20% blurred view images (Fig. 9) and 37% text recognition failure of 1080p view image (Fig. 13(b)) in practice, according to our empirical experience, it is better to obtain at least 6 view images of a POI (e.g., 3s) to ensure the reliability and efficiency of POI extraction. That means if there are 5 POIs around a user, the circle shoot will take 15s at least.

C. Floor-plan Images

EyeLoc generates high resolution indoor floor-plan images from Gaode Maps. Since the font of POI texts on floor-plan images is regular print format, the text recognition accuracy is close to 100%. The resolution of floor-plan image is set to 2560×1440. For error sensitivity estimation, we empirically set Δ_θ and R_e as 10° and 20 pixels. The threshold of text string similarity Δ_t is set as 50%.

V. Evaluation

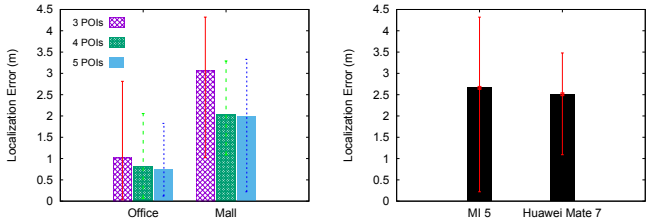
We evaluate EyeLoc with different smartphones (e.g., MI 5 and Huawei Mate 7) in office environment and two large shopping malls. The office environment is a 7m×9m office room as shown in Fig. 12. We print 6 shop signs such as NIKE on A4 size paper, then hang them on the wall or curtain in clockwise order. The area of each floor in two large shopping malls are 7,500 m^2 and 10,000 m^2 respectively. We invited two volunteers (Male, 20-30 years old) to complete all the experiments both in daytime and night. User 1 uses MI 5, User 2 uses Huawei Mate 7. The two users exhibit different habits as User 1 turns faster than User 2.



(a) The CDF of localization error in different environments

(b) The CDF of facing direction error in different environments.

Fig. 14: The reliability of EyeLoc.



(a) The influence of the number of observed POIs

(b) The localization error measured by different smartphones

Fig. 15: Different influencing factors.

In office environment, we uniformly split the office to 18 areas. Users stand at the center of each area. The ground truth is obtained through laser rangefinder. In two large shopping malls, we mainly select the positions near entrance, elevator and bathroom where users

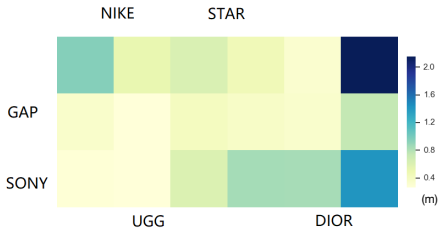


Fig. 16: The distribution of the localization error in office environment.

Fig. 17: Two experiment positions in two shopping malls.

have high localization demands. We evaluate 11 positions. For each position, we also use laser rangefinder to measure its ground truth. The minimum and maximum distances between the user and a POI are 2.26m and 29.2m.

A. The Accuracy of Localization and Heading Direction Estimation

First of all, we discuss the reliability of EyeLoc. As shown in Fig. 14(a) and Fig. 14(b), in the two large shopping malls, the median errors of localization and heading direction are 2.6m and 10.5°. Moreover, the 90-percentile errors of localization and facing direction increase to 4m and 20°. In the office environment, the 90-percentile errors of localization and facing direction are 1.1m and 8° which are much better than that in large shopping malls. There are two reasons. First, in office environment, we can precisely measure the POI coordinates on the floor-plan images. However, the POI coordinates suffer larger error due to the position mismatch between the text bounding boxes and the observed POI signs on the floor-plan images obtained from indoor map providers. Moreover, in office environment, we have only one POI in a view image in most cases. However, in large shopping malls, it is possible that several signs of the same POI may appear in a view image. It will incur error about the sightline estimation. EyeLoc fully considers these errors and designs countermeasures in both POI extraction and position matching. Given the area as large as 7,500m^2 and 10,000m^2, 4m and 20° is still relatively accurate for the most location based services.

1) The influence of the number of observed POIs: In position matching, EyeLoc utilizes the POI redundancy to mitigate the measurement errors of POI coordinates and EyeLoc sightline. Fig. 15(a) shows the relationship between the number of observed POIs and the localization error. We can see the average localization error decreases as the number of observed POIs increases. Specifically, the localization error decreases by 33.7%/34.9% when the number of observed POIs increases from 3 to 4/5 in large shopping malls respectively. This indicates the error mitigation approaches are effective for improving the localization accuracy. Moreover, 5 or more POIs cannot provides more useful information for improving the localization accuracy rather than 4 POIs.

2) The influence of POI distribution: We evaluate the influence of POI distribution on the localization error. Fig. 16 shows the localization error distribution of the 18 experiment positions given the positions of 6 POIs. The darker the color is, the higher the localization error is. We can see the localization error is getting higher when the distance between user's position and POIs is large. Moreover, the top left area is higher than its surrounding area, that is because it is close to the circle formed by several POIs. According to Section III-B, when the user's position and POIs tends to be on the same circle, the accurate localization will be hard to achieve. Moreover, we give two experiment positions in shopping mall 1 (Fig. 17(a)) and shopping mall 2 (Fig. 17(b)). The white areas are roads and yellow blocks are shops. The green points are the results of EyeLoc localization. The red points are the ground truths. The localization error of the position in shopping mall 1 is 1.56m, but that of the other one is 3.64m. From the POI distribution on the floor-plan images, we can see the large error in Fig. 17(b) is because the true position, GLORIA, AFU and MOFAN tend to on the same circle. In contrast, the position in Fig. 17 has more POIs which are close to it and uniformly distributed. Hence, the results suggest that the localization error is indeed related to the distribution of surrounding POIs, especially when the user's position and observed POIs are on the same circle. However, the situation only happens twice among total 29 positions. If the situation happens, we will ask the user to walk a short distance and relocalize himself/herself again.

3) The influence of smartphone hardware: We further evaluate the localization error by using different smartphones at the same positions in large shopping malls. The results are shown in Fig. 15(b). We can see the average localization error is 2.66m for MI 5 and 2.51m for Huawei Mate 7. Since User 2 turns slower than User 1, the more redundant view images make the localization error variance of Huawei smartphone is smaller than MI 5. Overall, EyeLoc works well on both smartphones and does not depends on any smartphone specific hardwares and parameters.

B. Processing Efficiency

The other important performance metric is the processing time, which is from the end of circle shoot to the user's position is shown on the screen. Since the view image processing dominates the overall processing time, EyeLoc have designed two approaches, namely outlier image filtering and sparse image processing. We

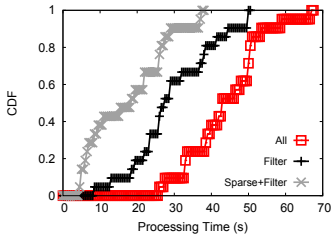


Fig. 18: The processing efficiency with outlier filtering and sparse processing.

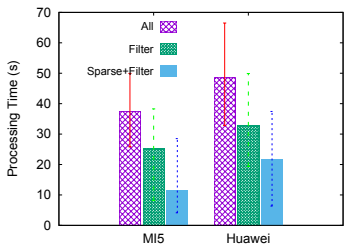


Fig. 19: The comparison of processing time on different smartphones.

further evaluate the processing time of three methods: "All" indicates we process all view images; "Filter" indicates we only process the view images after filtering the outliers. "Filter+Sparse" indicates we combine outlier filtering and sparse processing approaches. As shown in Fig. 18, we can see Filter+Sparse outperforms the other two methods and its median processing time is 18.2s which is 55.5% shorter than All. Moreover, the median processing time of Filter is 27.3s which is 36.2% shorter than All. That verifies both outlier filtering and sparse processing are effective to improve the processing efficiency. In the worst case, the processing time of Sparse+Filter is 37.4s. That means the user can obtain the localization result in no more than half minute after circle shoot. Fig. 19 further shows the processing time on different smartphones.

VI. Related Work

In recent years, many indoor localization systems are proposed. According to the type of physical information sources, we divide existing works into four categories.

Wi-Fi Signals Many indoor localization systems apply Wi-Fi signals. One approach is fingerprinting. The user's location is estimated by querying a measured fingerprint to a database constructed for the target area. Place Lab [2] use site survey to construct the fingerprint database, while LIFS [22] utilizes crowdsourcing. Another approach is to model the geometric relationship between Wi-Fi access points (AP) and the user through the propogation of Wi-Fi signals. EZ [3] uses the received signal strength (RSS) to estimate the signal propagation distance. SpinLoc [12] extracts the angle-of-arrival (AoA) of several APs to determine the location. ArrayTrack [20] uses the antenna array and phase shifts to obtain accurate AoA spectrum. Chronos [16] further refine the distance and AoA measurement methods to achieve high localization precision or adapt to COST Wi-Fi APs.

Visible Light Visible light positioning systems apply lamps (fluorescent and LED) as landmarks. The target senses light from landmarks and achieves self-localization. Luxapose [5] takes an image which contains several LEDs and calculate AoA. According to optical emission features of both LED and fluorescent, iLAMP [24] identifies a lamp's location from a pre-configured database by feature matching. It further combines camera image and inertial sensors to infer user's location. CELLI [18] develops a custom LED bulb which projects a large number of fine-grained light beams toward the service area. The light beam is modulated. SmartLight [8] uses LED array and a lens to form the light transmitter. The different PWM frequencies of LED lamps contain location information.

Others Magicol [13] and FollowMe [14] combine the geomagnetic field and user trajectories as the fingerprint to localize the user. Swadloon [4] and Guoguo [7] use acoustic signal to construct geometric models. Shenlong Wang, etc. [17] utilize the floor-plan image and a scene image to localize a user in large shopping malls. Based on edge, text and layout features of a scene image, they use Markov random field model to infer the camera pose on the floor-plan image. However, they need to search all possible positions which incurs huge computation complexity.

Compared with these methods, EyeLoc depends on neither pre-deployed infrastructure (i.e., Wi-Fi APs, custom lamps, acoustic speaker) nor pre-collected information (i.e., Wi-Fi fingerprint, lamp optical emission features, scene images, geomagnetic field). Moreover, EyeLoc does not require custom hardwares and can be implemented as a smartphone application.

VII. Conclusion

In this paper, we propose EyeLoc, a plug-and-play localization system for large shopping malls without suffering from the burden of system bootstrap and calibration. EyeLoc enables the smartphone to imitate the human self-localization behavior. After a user opens EyeLoc application, he/she carries out circle shoot, during which the smartphone continuously shoots view images. After that, EyeLoc automatically projects the user's position and heading direction on the floor-plan image. We implement EyeLoc as an Android application and evaluate its performance in various environments. The evaluation results show that the 90-percentile accuracy of localization and heading direction is 4m and 20°.

Acknowledgement

This study is supported in part by the NSFC program under Grant 61772446 and 61872081.

References

[1] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[2] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *Proceedings of ACM MobiSys*, 2005.

[3] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor localization without the pain. In *Proceedings of ACM MobiCom*, 2010.

[4] W. Huang, Y. Xiong, X.-Y. Li, H. Lin, X. Mao, P. Yang, and Y. Liu. Shake and walk: Acoustic direction finding and fine-grained indoor localization using smartphones. In *Proceedings of IEEE INFOCOM*, 2014.

[5] Y.-S. Kuo, P. Pannuto, K.-J. Hsiao, and P. Dutta. Luxapose: Indoor positioning with mobile phones and visible light. In *Proceedings of ACM MobiCom*, 2014.

[6] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye. Push the limit of wifi based localization for smartphones. In *Proceedings of ACM MobiCom*, 2012.

[7] K. Liu, X. Liu, and X. Li. Guoguo: Enabling fine-grained smartphone localization via acoustic anchors. *IEEE Transactions on Mobile Computing*, 15(5):1144–1156, 2016.

[8] S. Liu and T. He. Smartlight: Light-weight 3d indoor localization using a single led lamp. In *Proceedings of ACM SenSys*, 2017.

[9] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. Landmarc: Indoor location sensing using active rfid. *Wireless Networks*, 10(6):701–710, 2004.

[10] Q. Niu, M. Li, S. He, C. Gao, S. H. Gary Chan, and X. Luo. Resource-efficient and automated image-based indoor localization. *ACM Trans. Sen. Netw.*, 15(2):19:1–19:31, 2019.

[11] N. Roy, H. Wang, and R. Roy Choudhury. I am a smartphone and i can tell my user's walking direction. In *Proceedings of ACM MobiSys*, 2014.

[12] S. Sen, R. R. Choudhury, and S. Nelakuditi. Spinloc: Spin once to know your location. In *Proceedings of ACM HotMobile Workshop*, 2012.

[13] Y. Shu, C. Bo, G. Shen, C. Zhao, L. Li, and F. Zhao. Magicol: Indoor localization using pervasive magnetic field and opportunistic wifi sensing. *IEEE Journal on Selected Areas in Communications*, 33(7):1443–1457, 2015.

[14] Y. Shu, K. G. Shin, T. He, and J. Chen. Last-mile navigation using smartphones. In *Proceedings of ACM MobiCom*, 2015.

[15] R. Smith. An overview of the tesseract ocr engine. In *Proceedings of ICDAR*, 2007.

[16] D. Vasisht, S. Kumar, and D. Katabi. Decimeter-level localization with a single wifi access point. In *Proceedings of NSDI*, 2016.

[17] S. Wang, S. Fidler, and R. Urtasun. Lost shopping! monocular localization in large indoor spaces. In *Proceedings of ICCV*, 2015.

[18] Y.-L. Wei, C.-J. Huang, H.-M. Tsai, and K. C.-J. Lin. Celli: Indoor positioning using polarized sweeping light beams. In *Proceedings of ACM MobiSys*, 2017.

[19] H. Wu, Z. Mo, J. Tan, S. He, and S.-H. G. Chan. Efficient indoor localization based on geomagnetism. *ACM Trans. Sen. Netw.*, 15(4):42:1–42:25, 2019.

[20] J. Xiong and K. Jamieson. Arraytrack: A fine-grained indoor location system. In *Proceedings of NSDI*, 2013.

[21] Z. Yang, L. Jian, C. Wu, and Y. Liu. Beyond triangle inequality: Sifting noisy and outlier distance measurements for localization. *ACM Trans. Sen. Netw.*, 9(2):26:1–26:20, 2013.

[22] Z. Yang, C. Wu, and Y. Liu. Locating in fingerprint space: wireless indoor localization with little human intervention. In *Proceedings of ACM MobiCom*, 2012.

[23] P. Zhou, M. Li, and G. Shen. Use it free: instantly knowing your phone attitude. In *Proceedings of ACM MobiCom*, 2014.

[24] S. Zhu and X. Zhang. Enabling high-precision visible light localization in today's buildings. In *Proceedings of ACM MobiSys*, 2017.