PerLoc: Enabling Infrastructure-Free Indoor Localization with Perspective Projection

Puchun Feng, Lan Zhang, Kebin Liu, Yunhao Liu
School of Software and TNLIST, Tsinghua University
Email: fengpc12@mails.tsinghua.edu.cn, {zhanglan, kebin, yunhao}@greenorbs.com

***Abstract*—With the rapid development of mobile applications, there is an urgent need for highly efficient indoor localization service. Dedicated systems achieve good accuracy at the cost of deploying special hardware. Fingerprint-based methods avoid maintaining the expensive infrastructure but suffer from intensive labor for site-survey and poor robustness. In this paper, we present PerLoc, an infrastructure-free localization system which leverages rich vision features in indoor environment with high efficiency and accuracy. PerLoc makes use of binocular ranging technique to calculate the depth of a feature point and then figures out its geographical coordinates to build up reference point database, for which we design a filtering scheme to keep the database efficient in storage and search delay. During the localization stage, users simply take a photo of surroundings and feature points are extracted automatically as input for search scheme. Then a fast two-stage search scheme is proposed to find the nearest neighbors of query feature points in reference point database. Based on the perspective projection model, we inversely calculate users' geographical location in realtime. We implement the proposed localization system on commercial smartphones as well as laptops and conduct extensive experiments. PerLoc achieves $1.76m$ of average error in office environment, and $2.2m$ of average error in shopping mall.**

I. Introduction

Worldwide user base of smartphone hit 1 billion in 2012 and the current number of apps available in Apple's AppStore exceeds 700,000, meanwhile there is about the same number of live Android apps in the market. Accurate localization service is a fundamental service for a significant portion of these applications. While the outdoor localization has been solved by GPS, localization in indoor environment remains an open question, leading to largely restricted adoption of many apps for indoor scenarios. Although numerous research studies as well as industrial efforts have been devoted to this problem, heretofore there still lacks a feasible and widely applied approach for indoor localization. Existing work can be generally classified into two categories. One category includes many dedicated systems that require deploying infrastructure [1] [2] in priori or leveraging specialized hardware [3] [4]. These methods have achieved high accuracy but can hardly be applied on smartphones whose demands are stringent. Besides, they suffer from expensive infrastructure resulting in the difficulty of scaling to pervasive usage. The other category of solutions is infrastructure-free. Most of extant infrastructure-free approaches use fingerprints to index locations, where they are collected, and build fingerprint databases. On the localization stage, user's smartphone senses fingerprints which are then compared with ones stored in database. Then the location label of the best matched fingerprint is returned to the user. The most commonly applied fingerprints are WiFi signals from prevalent wireless access points. These methods, however, still have some limitations for practical applications. First, they need exhausted survey of wireless fingerprints on all locations, which is very costly. To fulfill this task, crowdsourcing methods [5] are considered in recent works. Second, these schemes suffer a lot from the dynamics of signal distribution. For example, the displacement of an access point can significantly degrade their performance. Finally, although reasonable accuracy can be achieved, fingerprint-based approaches can have significant errors due to one of their fundamental limits, distinct locations with similar signatures. Some researchers propose to use image matching which determines the user's location by searching the most similar images in database and then returns the location where the matched image is surveyed. These methods incur large amounts of storage cost to maintain image database with very high delay for image searching. Furthermore, they can only provide room-level accuracy due to the sparse sampling.

To address these issues, we present PerLoc, a novel type of infrastructure-free indoor localization system, which explores the rich and stable vision features of the indoor environment and makes use of the geographical information of these vision features to inversely calculate user's location. Site-survey of PerLoc is conducted only at a few locations by taking photos of indoor environment with two cameras and then we extract vision features from photos. We leverage the binocular ranging technique to calculate the depths of vision features, which are used to figure out vision features' geographical coordinates. We propose a two-stage, cluster-based search scheme to fast find the nearest neighbors of user's query feature points. On the localization stage, according to perspective projection model we avail of query features' nearest neighbors to inversely calculate user's geographical location in realtime and with high accuracy. As an infrastructure-free indoor localization system, PerLoc avoids the limitations of existing fingerprinting schemes and exhibits the following advantages: 1) Lightweight and easy to operate site-survey. Different from fingerprinting, our method does not require an exhaustive site-survey, with only a few locations surveyed for generating reference points. 2) Robust. The vision features leveraged in this work are highly distinctive, invariant to scale, rotation, distortion, addition of noise, change in viewpoint, and change in illumination and thus avoid the impact of dynamic signal distribution. 3) Highly

accurate and realtime. PerLoc achieves $1.76m$ of average error in office environment, and $2.2m$ of average error in shopping mall. In office environment PerLoc only takes no more than $1.5s$ for 90% queries, $1s$ for 90% in shopping mall and the average time is no more than $1.5s$ for both.

The major contributions of this work are summarized as follows:

- We present a new framework leveraging vision features to achieve highly efficient indoor localization. Based on the perspective projection model, we inversely calculate the geographical location of a user in realtime.
- We enhance the performance of inverse localization by the optimization of the error estimate function, which appends a vote procedure to select the final result.
- We propose an integrated system design including novel approach for vision based site-survey, fast search scheme for the query feature's nearest neighbor and inversely calculation of user's location.
- We implement the proposed scheme on both smartphone and PC, and also conduct extensive experiments to verify the effectiveness of our approach.

The rest of the paper is organized as follows. Section 2 introduces the system architecture. Section 3 presents the consideration and design of site-survey approach. Section 4 describes the two-stage search scheme. Section 5 elaborates the algorithm of inverse localization and optimization about it. The performance is evaluated in Section 6 and related work is summarized in Section 7. Finally we concludes this paper in Section 8.

II. System Overview

In this section, we provide a brief overview of PerLoc architecture as shown in Figure 1. PerLoc firstly builds up reference point database which stores the geographical information of reference points together with vision descriptors extracted by the site survey component. When a user sends query feature points extracted by the querier component to the server, the search component finds the nearest neighbors of query features by two-stage search scheme. At last, the localization component inversely calculates the user's location and sends the result to the user.

PerLoc consists of four components:

(1)Querier Component: This component runs on smartphone as an application for users. It extracts the feature points from the query photo taken by a user, and generates a descriptor for each feature point. But we don't need all feature points, for the reason of saving communication cost and filtering out the unstable feature points, *i.e.* features not on building structures or features on people, which would decrease accuracy of localization.

(2)Site Survey Component: Site survey component builds up the specific reference point database for a building. We calculate the geographical location of a reference point according to its depth which is figured out by binocular ranging. Binocular ranging makes use of the disparity of one feature point to calculate its depth. In order to obtain the disparity

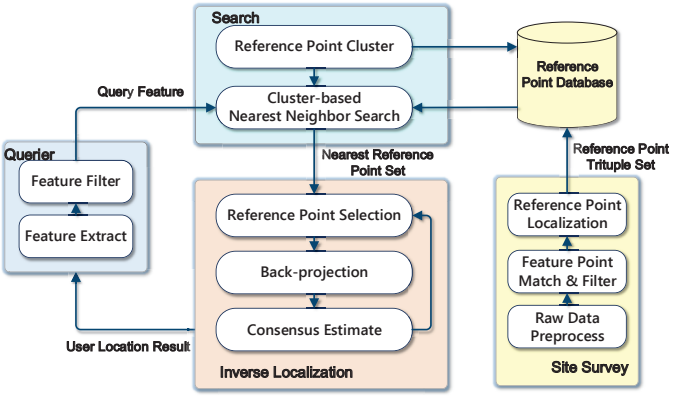


Fig. 1. PerLoc system architecture.

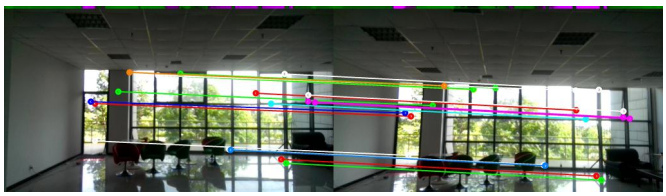


Fig. 2. Pair-photo with feature point matches.

of feature points, we firstly take pair-photos(Figure 2) for the building on the predefined survey track. Then this component extracts feature points from pair-photos. It matches the feature point in the left photo to the most similar feature point in the right photo to calculate disparity of this feature. However, some matches are not similar enough or not reasonable, *e.g.* one feature on the floor matches to one on the ceiling though their descriptors are similar. These bad matches should be thrown away because they may impact the calculation of geographical position of reference point. Then we indicate the geographical coordinates of feature point according to its depth and pixel coordinates. We combine the descriptor of a feature point with its geographical coordinates to form a reference point and store all of them into reference point database.

(3)Search Component:

Given a localization request, the search module will find the nearest neighbor for each query feature descriptor using two-stage cluster-based search algorithm. The cluster-based search algorithm makes use of feature point clusters generated by reference point cluster component. One cluster consists of reference points whose feature descriptors are near to each other in Euclidean distance. In one cluster, we choose a reference point to represent all points in it and this representative can be treated as the centroid of cluster.

(4)Inverse Localization Component: This component inversely calculates user's location as soon as the search component outputs the nearest neighbor set of query feature points. Firstly, we choose the trine of match from search component's outputs. Each trine contains three matches between query points and reference points. We sort these trines according to the similarity of matches measured by the distance between t-

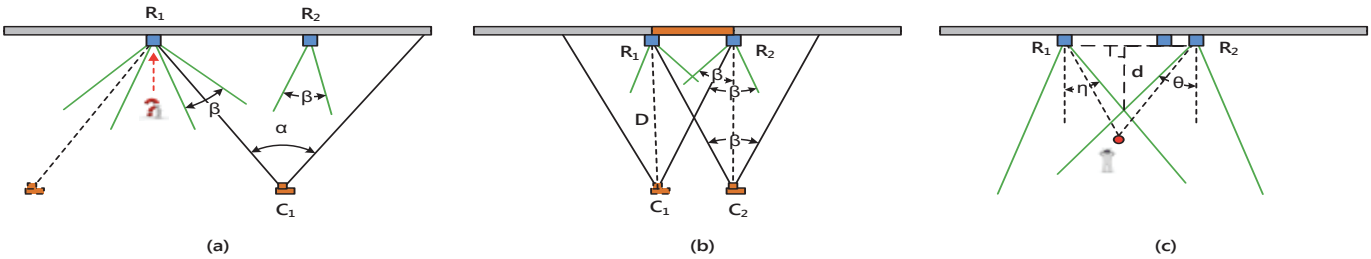


Fig. 3. The observations of site-survey.

wo feature points. The trine whose matches have high similarity will get high priority. We calculate the geographical location of user by solving the location determination problem(LDP). As we figure out a location, we estimate the error of location by calculating the consensus of it in its nearest neighbor set. The location with low consensus is possible to be inaccurate, so we discard it and start another round of inverse calculation. This process stops until the number of rounds exceeds the threshold. At last, this component returns the most accurate result of user's location.

III. Site Survey Approach

The site-survey stage extracts and selects a set of reference points in the indoor environment then builds up a reference point database. Instead of collecting wireless fingerprints at all coordinates, in PerLoc we just extract feature points at only a few locations for the reason that at one shooting point we are able to collect many feature points as long as no obstacle lies between us and building structure. Each reference point stored in the database is denoted as a trituple $R = (v, c, a)$ where v is a vector of vision features which we refer to as the *feature descriptor* in this work, c represents the geographical coordinates of the reference point and a denotes the visible orientation of this reference point. In this work, we leverage the state-of-the-art feature detector SURF [6], a performant scale- and rotation-invariant interest point detector and descriptor, which is inspired by the SIFT descriptor [7], and is faster than SIFT. With this typical setting, a SURF feature descriptor can be represented as a vector with 64 dimensions.

A. Observations of Site Survey

We are aiming to collect reference points associated with the building structure for the reason that these features are relatively stable. A straight-forward solution is to take pictures at consecutive locations, as shown in Figure 3(a). Let α denote the view angle of the camera *i.e.* the photo can cover reference points in a range of α degree. This method ensures that all reference points be recorded from one direction. However, there is another factor which should be considered, the visible orientation of the reference points which is denoted as β. Note that the SURF features retain high matching accuracy (80%) out to a 30 degree change in viewpoint. In other words, if the users look at the same reference point from a direction that differs from that in site-survey stage for more than 15 degree, the reference point could be unrecognizable. Therefore, in this work, we set $beta$ to 30 degree. As shown in Figure 3(a), while standing in the shadow region the user will have difficulties in identifying reference points in front.

There is no need to guarantee that all reference points can be identified from all directions, thus our design goal is to approximately ensure a proper view angle in front of each reference point, which means the reference point can be recognized from the direction orthogonal to its background building structure or even tolerates small skews in view angle. As β is usually smaller than α, we first consider α as wide as β and set the distance between two shooting points to $D \cdot \tan(\frac{\beta}{2})$ as shown in Figure 3(b). D denotes the distance from the shooting point to the building structures. All reference points are covered by at least two pictures taken from adjacent points. The orange range of background is sampled at both C_1 and C_2 and the visible orientation of each reference point is the superposition of the visible orientations in two samples. By geometric analysis, all reference points in the overlapping range can be recognized from the direction orthogonal to the background with an angle skew less than $\frac{\beta}{2}$ to both clockwise and anticlockwise. In practical usage, since α is usually larger than β and we can take multiple pictures in different directions in one location, we have sufficient redundancies in the quantity of reference points to ensure the accuracy of query.

B. Reference Points Geographical Coordinates Calculation

Based on previous observations, we make a survey track according to the building structure. We annotate every shooting point in the track with its coordinates in the local coordinate system of the building. We take a pair-photo, including a left and a right photo, at every shooting point. We use two cameras to take pair-photo, the left camera is $d_{cam}mm$ apart from the right camera.

1) Depth Calculation: Firstly, we will calculate the depth of feature point extracted from pair-photo using binocular ranging technique. Figure 4 shows the geometry model of calculation for reference point's depth. Optical center is the center of the perspective projection, the optical axis is perpendicular to the image plane piercing through the principal point on the plane. Without loss of generality, the coordinate system is selected such that its origin is located at the optical center of the left camera and the Z-axis is collinear to the optical axis, as shown in Figure 4. $P\ (X, Y, Z)^T$ is a geographical coordinate of one feature point and the two corresponding projected pixels on

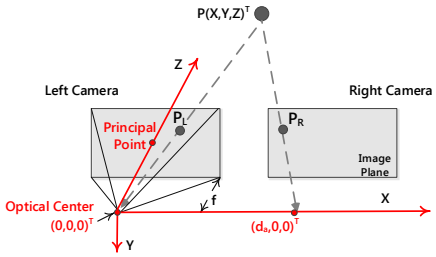


Fig. 4. Model of reference point's depth calculation.

image planes are $P_L(x_L, y_L)$ and $P_R(x_R, y_R)$, where x_L, x_R, y_L, y_R denote pixel coordinates. Let the principal point of left camera be $(o_x, o_y)^T$. The projection of P on the image plane of the left camera is

$$x_L = \frac{X \cdot f}{Z} + o_x, y_L = \frac{Y \cdot f}{Z} + o_y. \tag{1}$$

Expressed in matrix:

$$\lambda_L \begin{pmatrix} x_L \\ y_L \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}. \tag{2}$$

λ_L is a scaling factor. Let $K = \begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{pmatrix}$, when the origin of the pixel coordinate system corresponds to the principal point, $(o_x, o_y)^T = (0, 0)^T$. The right camera is located on the X axis whose optical center is $(d_{cam}, 0, 0)$. Because both cameras are identical, we have:

$$\lambda_R \begin{pmatrix} x_R \\ y_R \\ 1 \end{pmatrix} = K \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} - K \begin{pmatrix} d_{cam} \\ 0 \\ 0 \end{pmatrix}. \tag{3}$$

The depth Z of P can be expressed combining with coordinates of P_L and P_R,

$$\begin{pmatrix} x_R \\ y_R \end{pmatrix} = \begin{pmatrix} x_L - \frac{f \cdot d_{cam}}{Z} \\ y_L \end{pmatrix}, \tag{4}$$

$$Z = \frac{f \cdot d_{cam}}{Disparity}, \tag{5}$$

where $Disparity = x_L - x_R$.

2) Geographical Coordinates Calculation: Figure 5 shows the geometry model of reference point geographical coordinates calculation. As we figure out the depth Z of reference point P, we are able to calculate the coordinate X and Y of P.

$$X = Z \cdot tan\theta \tag{6}$$

where θ is the angle between l_0 and l_3, it is also as wide as the angle between l_3 and l_2. θ is proportional to L, which is horizontal component of pixel distance J from P_I to the principal point, P_I is perspective projection of point P on the image plane. The view angle α of camera is usually constant, so

$$\theta = \frac{L \cdot \alpha}{w_{photo}} \tag{7}$$

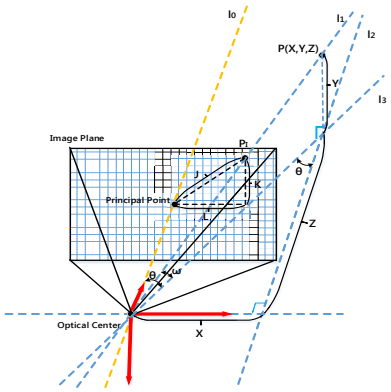


Fig. 5. Model of reference point geographical coordinates calculation.

where w_{photo} is the width of the photo in pixel. Similarly, we can calculate the angle ω which is between l_3 and l_2. According to the height of optical center h_{oc}, we figure out

$$Y = h_{oc} \pm (\sqrt{X^2 + Z^2} \cdot tan\omega) \tag{8}$$

which takes a plus sign when P_I is above the principal point, minus sign when beneath.

C. Reference Points Filtering

High redundance of reference points is prone to incur high storage and search overhead. Additionally, indistinctive reference points may lead to error localization results. Therefore, it is necessary to filter the reference points to retain the most distinctive ones while ensuring the selected reference points evenly distributed in the indoor environment with desired density for localization.

To address this issue, we present an iterative filtering scheme. At the very beginning, all reference points are partitioned into different clusters according to their feature descriptors. Then reference points within each cluster are ordered by their *neighbor density*, defined as the number of neighbors of one reference point with spacial distance less than r.

In each iteration, we select the largest cluster and check the reference point with the highest neighbor density. Then the neighbor densities of reference points in the cluster are updated as well as their orders. If the neighbor densities of all points in a cluster are all lower than the the pre-specified threshold D_{min}, this cluster is marked as *done* and will not be processed in the next iteration. The above operations repeat until all clusters are marked as *done*.

As is shown in Figure 3(c), the user can recognize both reference points R_1 and R_2 while standing in the shadow area. As is discussed in prior subsection, the angle η and θ are within the range of $[\frac{\beta}{2}, \beta]$, thus the distance between R_1 and R_2 which is calculated as $d \cdot (\tan\eta + \tan\theta)$ is within the range of $[d \cdot \tan\beta, 2 \cdot d \cdot \tan\beta]$. Here we assume that users will stand 2 meters away from the building structures at most of the times, thus $d \leq 2$. Then the distance between R_1 and R_2 should be less than $d \cdot \tan\beta = \frac{2}{\sqrt{3}}$. We need at least three

reference points to inversely localize the user which will be discussed in following sections, so we require that at least one reference point exists between R_1 and R_2. To satisfy this condition, the distance from one reference point to its nearest neighbor should be less than $\frac{d}{2} \cdot \tan\beta = \frac{1}{\sqrt{3}}$. This setting is just a sufficient but not necessary condition for our localization scheme since users can change their view point to get more reference points. Then we set r equal to $\frac{d}{2} \cdot \tan\beta = \frac{1}{\sqrt{3}}$ and D_{min} equal to 1.

IV. Search Scheme

This section describes the reference point search scheme, which matches query feature points to reference points for the purpose of acquiring the nearest neighbors. We transform the matching problem to the well-known Nearest Neighbor (NN) search problem. The distance between two feature points equals the Euclidean distance of their descriptors. We also use this distance to measure the similarity of a match between query point and reference point and that a match has short distance means a good match with high similarity, and vice versa.

In fact, exhaustive search for the NN in a large database is costly for both space and time. Many data organization and search schemes has been proposed to speed up the process, we propose a fast two-stage search scheme based on cluster [8]. Both the visible orientation and feature descriptor of the point are considered in our method. Our two-stage search scheme includes preprocessing stage and search stage.

A. Preprocessing Stage

In the preprocessing stage, we firstly partition all reference points into b bins according to their visible orientation. Though reference point may be replicated in bins that overlap with its visible orientation, we just search the NN of the query feature point in only one bin according to its shooting angle. In each bin, we randomly choose $\sqrt{N}$ reference points as the representatives of clusters. Then we put every reference point into the cluster whose representative is the closest to it. At last, we acquire $\sqrt{N}$ clusters, each of which has a unique representative. The cluster algorithm is described in Algorithm 1.

For parameter b we make a tradeoff between the space and the search time. As b increases, more bins are introduced and the angle range of each bin decreases. Thus the visible orientation of each reference point will span more bins and have more replications. Since the angle range of each bin gets wider, the number of reference points in each bin decreases, the search time decreases as well. Without loss of generality, we set b equal to 30 degree.

B. Search Stage

In the search stage, in order to find descriptor Q's nearest neighbors in its visible orientation bin, we propose a 2-round search algorithm. In the first round, we calculate the distance between Q and each representative. We denote the closest representative as T. It is supposed Q's NN belongs to T's cluster. Therefore, we look for Q's NN in T's cluster in the second round. We compute the distances between Q and each feature point in T's cluster. Finally, we obtain K closest feature points as Q's top-K NN.

Algorithm 1 Clustering Descriptors

Input: reference point set S,set size N
Output: representative set R, cluster hash map $C\langle Representative, Cluster\rangle$

$R \leftarrow null$, $C \leftarrow null$
while $|R| < \sqrt{N}$ **do**
  Randomly choose a reference point d from S
  if $d \notin R$ **then**
    $R \leftarrow R \cup d$
  else
    continue
  end if
end while
for all $r \in R$ **do**
  $w \leftarrow \phi$
  $C.put(r, w)$
end for
for all $d \in S$ **do**
  $f \leftarrow$ the cluster whose representative is closest to d
  $f.add(d)$
end for
return C, R

C. Complexity Analysis

In the preprocessing stage, assume there are b bins and each bin has N reference points averagely. With respect to one bin, we will compare N reference points to $\sqrt{N}$ representatives, *i.e.* we will do $b \times N \times \sqrt{N}$ comparisons in all. At query time, we compare the query feature descriptor Q to representatives in the specific bin determined by the orientation of user's camera. Q is firstly compared to $\sqrt{N}$ representatives to find the nearest cluster f. Then Q is compared to each reference point in f. On average, each cluster contains $\sqrt{N}$ reference points. Thus, for one query feature point Q, we need to do a total of $2\sqrt{N}$ comparisons.

V. Inverse Localization

A. Location Determination Problem

In this section, we form our localization problem as the Location Determination Problem (LDP) [9]. The definition of a traditional LDP problem is as follows. Given a photo depicting a set of n reference points with known locations, we determine the geographical coordinates of the place where the photo was taken. It has been proved that when n equals to 3, the the problem can be solved [9]. We derive the orientation of camera with onboard sensors of the smart phone which helps to facilitate the localization process. We assume that the optical axis of the camera pierces the image plane on the *principal point* and the *focal length* of the camera is known. As shown in Figure 6, the user takes a photo of

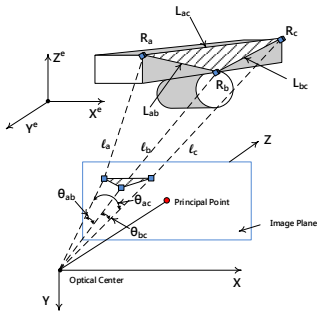


Fig. 6. Determination of the location of user based on reference points.

the scene in front of him/her and three references points, namely, $R_a(X_a, Y_a, Z_a)$, $R_b(X_b, Y_b, Z_b)$ and $R_c(X_c, Y_c, Z_c)$ are projected to the image plane. Through the reference point search scheme, we have obtained the correspondences between image points and reference points. According to the distance from the image point projected by R_a to the the point projected by R_b, we can easily derive θ_{ab}, the angle between the line from the *optical center* to R_a and the line from the *optical center* to R_b. We can also derive θ_{bc} and θ_{ac} in the same way. Let L_{ab}, L_{bc} and L_{ac} denote the distances between three reference points respectively, and l_a, l_b, l_c denote the distances from each reference point to the *optical center*. We have the following equation group according to the law of cosine:

$$\begin{cases} (L_{ab})^2 = (l_a)^2 + (l_b)^2 - 2l_a \cdot l_b \cdot \cos(\theta_{ab}) \\ (L_{bc})^2 = (l_b)^2 + (l_c)^2 - 2l_b \cdot l_c \cdot \cos(\theta_{bc}) \\ (L_{ac})^2 = (l_a)^2 + (l_c)^2 - 2l_a \cdot l_c \cdot \cos(\theta_{ac}) \end{cases} \quad (9)$$

The above equation group may have multiple solutions, we make use of the technique proposed in [9] to solve it and use the orientation of the camera to eliminate infeasible solutions. Every solution includes distances from three reference points to the *optical center*.

We use the distance between query feature point and its nearest neighbor of reference point to measure the similarity of the match mentioned before. Each time, we select three matches with the highest similarity from the result set T of that searching stage to form a trine.

Based on the solution of equation group 9, we are able to figure out the geographical location of the *optical center* $CP(X, Y, Z)$, *i.e.* user's geographical location, by solving another equation group 10.

$$\begin{cases} \sqrt{(X - X_a)^2 + (Y - Y_a)^2 + (Z - Z_a)^2} = l_a \\ \sqrt{(X - X_b)^2 + (Y - Y_b)^2 + (Z - Z_b)^2} = l_b \\ \sqrt{(X - X_c)^2 + (Y - Y_c)^2 + (Z - Z_c)^2} = l_c \end{cases} \quad (10)$$

Equation group 10 describes the distances from CP to three reference points. This equation group may have multiple solutions as well. Every solution will be treated without distinction to estimate its error.

We take the concept of "consensus" proposed by [9] to estimate error of each solution. With respect to the solution S_0, one reference point r from T reaches a consensus with S_0 if r passes the estimate check function $Func$. $Func$ can be designed in a variety of forms. Our estimate check function $Func1$ is designed as follows: $Func1$ computes reference point r's pixel coordinates in the picture which takes S_0 as the optical center. If the distance between r's newly computed coordinates and its original coordinates does not exceed threshold t, then r reaches a consensus with S_0. We put reference points which reach consensus with S_0 into consensus set C_{S_0}. If the ratio $\frac{|C_{S_0}|}{|T|}$ exceeds the threshold w, then S_0 will be considered as one candidate solution and added to set F. After every solution has been processed, we are able to figure out the final result of CP's location from set F by selecting the candidate with highest consensus.

B. Optimization of Estimate Check Function

Aiming at enhancing localization performance of $Func1$, we refine it to form another estimate check function $Func2$ by appending a post-process after $Func1$. When we obtain F, rather than taking the candidate with highest consensus as the final result, we select the candidate through the vote. For one candidate c, a reference point r in T will vote for c if the distance from r to c is not longer than threshold d_{th}. We discard the candidates with fewer votes and keep K candidates with more votes. From these K candidates, we pick the one derived from the trine whose triangle area is the largest among them as the final result.

$Func2$'s post-process is inspired by this phenomena: if three reference points in one trine are close to each other, then the solution of Equation group 9 is more prone to error. If a trine has a larger area, the distances between three reference points in it are comparatively longer, which reduces the chance of error prone.

VI. Implementation and Evaluation

A. System Implementation

We have implemented PerLoc both on Android smartphone and laptop. The user application is implemented on HTC NEW ONE X with azimuth sensor and WiFi , as well as the site-survey tool application. The server side of PerLoc is implemented on ASUS A43S laptop installed Win7 with four 2.20GHz Intel Core i7 processors, 8GB of DDR2-memory.

For the software part of client application, the code for feature extractor module is implemented using C++ and OpenCV library for Android, the remaining parts are implemented in JAVA , the client application runs on the Android Operating System. The site-survey tool application also runs on Android platform. The server side is implemented using C++ and JAVA.

B. Case Study I: Office Environment

1) Site Survey in Office Environment: As we have the floor plan of the experimental office environment with $60m \times 28m$ area, we annotate each shooting point on the floor plan with its relative geographical coordinates. Figure 7 shows the floor plan of the office, the yellow star is the original point of the office coordinate system, the red line marks the survey track and white dots mark the shooting points, the distance between

every two adjacent shooting points is $1.6m$. On each shooting point, we will take pair-photos from two directions, if the shooting point is on the horizontal segment of green line we take photos from north and south, on the vertical segment we take photos from east and west.

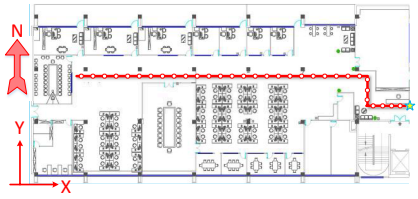


Fig. 7. Site survey track in office.

After processing the photos and calculate the geographical locations of reference points, we are able to project the reference points denoted by red points on the floor plan as shown in Figure 8. We have filtered out some reference points, the remaining reference points are stable, *i.e.* they are located on the building structure like walls, windows and so on.

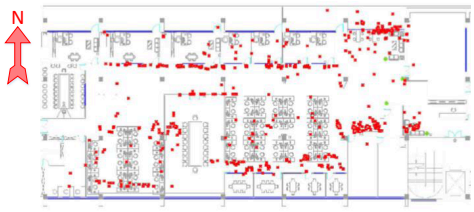


Fig. 8. Project filtered reference points on floor plan.

2) Impact of Reference Point Database Size: In the stage of building up reference point database, the tighter parameters for reference points filtering the less reference points remain, thus we can obtain several reference point databases with different sizes by set different parameters. We study the clustering cost of time and space for a range of sizes of reference point database, Table I shows the results. As analysed in section IV-C, having more reference points results in a longer clustering process, more clusters and larger space, and conversely the less overhead in time and space. Figure 9(a)

ID	Ref points	Clusters	Time(ms)	Average cluster size(KB)
1	2718	52	603	31.9
2	2088	46	371	27.6
3	1279	36	362	21.75
4	816	29	274	17.3

TABLE I
IMPACT OF REFERENCE POINT DATABASE SIZE ON CLUSTERING STAGE FOR OFFICE ENVIRONMENT.

shows the localization error(LE) run on four reference point databases of office environment. Obviously, database 1 which has the largest number of reference points achieves the lowest average error(AE) $1.76m$ among four databases. Database 4 which has the least number of reference points performs a low localization accuracy, its AE is $19.4m$. The AE of database 2 is $10.4m$ which is close to database 3. This result tells us that there is a tradeoff between overhead and localization accuracy, *i.e.* the database with larger size will have more accurate localization results, however, create more overhead and the database with smaller size will cause less overhead but acquire worse localization results. Taking the office environment into account, we are convinced that the size of database for the building whose area is about $1600m^2$ should be almost 2000 reference points.

Figure 9(b) shows the CDF of the LE run on database 1. On database 1, we can locate the user with an accuracy of $1.58m$ for 50% and $3.56m$ for 91%. Figure 9(c) shows the localization time(LT) run on four office reference point databases, the time of localization process is no more than $2.2s$. With respect to database 1, the localization time is $0.5s$ for 50% and $1.5s$ for 90%, which is fast enough to achieve real time localization.

3) Impact of Check Function $Func$: This section gives an inspection on how different check function $Func$ affect the performance of PerLoc. We use $Func_1$ and $Func_2$ respectively to make accuracy experiments on database 1, the results are shown in Figure 9(d). Compared to $Func1$ whose LE is $5.1m$ by 91%, $Func2$ improves the localization accuracy notably, whose LE is $3.9m$ by 91%.

C. Case Study II: Shopping Mall

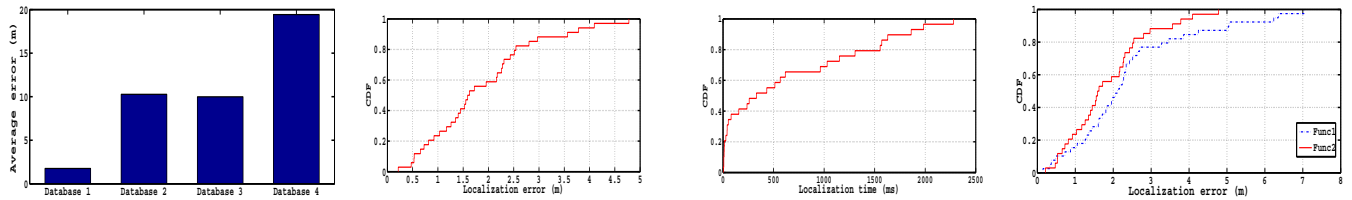
1) Site Survey in Shopping Mall: We do a site survey for the shopping mall without floor plan, it means that we have to manually mark relative coordinates. Figure 10 shows the shooting points by purple diamonds, origin point by red star and filtered reference points by blue crosses. The distance between every adjacent two shooting points is $1.2m$. The shopping mall with $130m \times 13m$ area consists of three little plazas which are connected by two passages. We can tell the shopping mall's structure from the Figure 10 which shows the projected reference points.

2) Reproduction of Results in Previous Work: In this section, we study the performance of PerLoc in the scenario of shopping mall. Table II shows the clustering cost of time and space for a range of size of reference point database in the shopping mall where we conduct experiments on three databases with different sizes.

ID	Ref points	Clusters	Time(ms)	Average cluster size(KB)
1	2465	50	625	30.92
2	1739	42	516	26.08
3	1327	36	453	23.36

TABLE II
IMPACT OF REFERENCE POINT DATABASE SIZE ON CLUSTERING STAGE FOR SHOPPING MALL.

Figure 11 shows the localization performance in shopping mall reference point database, which is similar to the results in office environment. As shown in Figure 11(a) the AE for database 1 is $2.2m$. The LE run on database1 is $3.4m$ for 90%,



(a) Average LE (b) CDF of LE (c) CDF of LT (d) CDF of LE

Fig. 9. Localization performance in office environment reference database: (a)Average LE of 4 reference point databases. (b)CDF of LE run on database 1. (c)CDF of localization time on database 1.(d)CDF of LE for $Func1$ and $Func2$ in office environment.

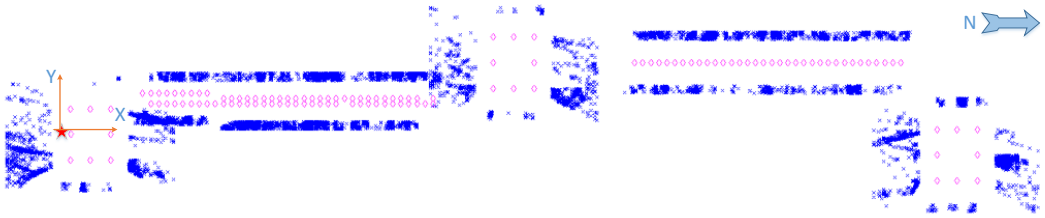
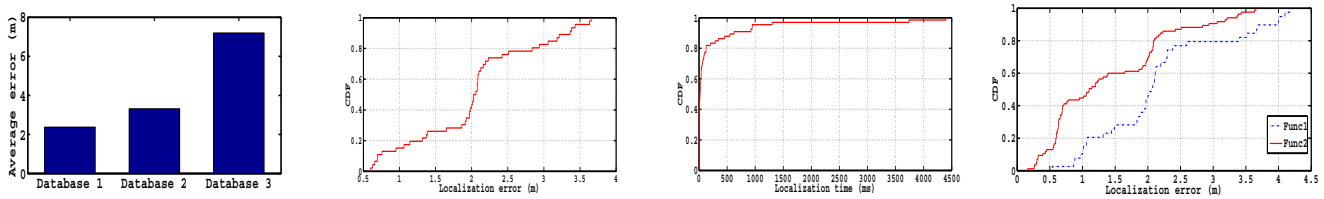


Fig. 10. Filtered reference points of shopping mall.



(a) Average LE (b) CDF of LE (c) CDF of LT (d) CDF of LE

Fig. 11. Localization performance in shopping mall reference points database: (a)Average LE of 3 reference point databases. (b)CDF of LE on database 1. (c)CDF of localization time on database 1.(d)CDF of LE for $Func1$ and $Func2$ in shopping mall.

the LT is $1s$ for 90% as shown in Figure 11(b)(c). $Func2$ improves the localization accuracy by $0.7m$ for 90% queries in shopping mall shown in Figure 11(d).

VII. Related work

There are many dedicated indoor localization systems with specialized hardware, *e.g.* sensors and RFID, which can achieve high accuracy, but are not scalable. Many application scenarios of mobile social networks are indoor. One popular line of mobile handset indoor localization is fingerprinting. These approaches collect fingerprints with known locations and compare the observed measurement at unknown locations with all known fingerprints to find the best match. Some systems depend on fingerprints of wireless signals to achieve room-level user localization. RADAR [10] proposes a radio-frequency (RF) based system for locating and tracking users inside buildings. Horus [11] designs a WLAN location determination system which achieves meter-level localization.

OIL [13] uses Voronoi regions for conveying uncertainty and reasoning about gaps in coverage, and a clustering method for identifying potentially erroneous user data to facilitate rapid coverage while maintaining positioning accuracy. EZ [14] exploits the RSSI of indoor APs to estimate the user's location with no pre-deployment effort and yield a median localization error of 2m and 7m.

[16] presents a solution for achieving high speed 3D continuous localization for a pair of phones using two microphones with one speaker on a phone. This approach uses acoustic cues based on time-of-arrival and power level with assistance of accelerometers and digital compasses. [17] obtains acoustic ranging estimates among peer phones and maps phones' locations jointly against WiFi signature map subject to ranging constraints to reduce the significant errors of WiFi-based method. Virtual Compass [18] designs a peer-based relative positioning system on commodity mobile handhelds to determine proximity. It uses multiple radios to detect nearby mobile devices and places them in a two-dimensional plane. It uses adaptive scanning and out-of-band coordination to explore trade-offs between energy consumption and the latency in detecting movement.

Structure from Motion(SfM) reconstruction approaches enables the creation of large scale 3D models of urban scenes [19], which can be used in image-based localization through computing 2D-to-3D correspondences. To improve the performance of 2D-to-3D matching methods, [19] derives a visual vocabulary quantization based matching framework and a prioritized correspondence search. Compared with SfM, PerLoc builds up the reference points database by binocular ranging which combines the position of shooting point with feature

point 3D coordinates computation. In this way, PerLoc reduces the error of 3D structures of indoor environment.

[20] presents a pipeline system to perform image-based localization of mobile devices. A 3D geo-referenced image database is generated by making use of a specific human operated backpack. It matches SIFT feature extracted from query images against the 3D point cloud, combining user's cell phone sensor information like pitch and roll, to recover the user's pose.

Some vison-based localization works [21] [22]compares a test image against a database of pre-captured benchmark images, finds the "closest" match and uses its location. [23] combines photos and gyroscope on smartphone to localize users by measuring users' relative positions to physical objects. [24] allows users to locate remote objects by a few photos from different known locations.

VIII. Conclusion

In this work, we study the problem of building a practical system which provides realtime and accurate indoor localization service for smart phone users. While many innovative proposals have been presented in the literature, we take advantage of the SURF vision feature and perspective projection model to build reference point database and to perform indoor localization without deploying any infrastructure. Our approach is simple but works effectively to provide localization service in indoor environment with reasonable accuracy. We further improve the accuracy using enhanced estimate check function $Func2$. Through all processes of PerLoc, there is not much expenditure in time and cost for site-survey or running the system. We are convinced that our proposal can suggest a new approach combining computer vision technique with indoor localization research.

IX. Acknowledgements

This research is supported by NSF China Major Program under Grant No.61190110. We thank all the reviewers for their valuable comments and helpful suggestions.

References

[1] G. Borriello, A. Liu, T. Offer, C. Palistrant, and R. Sharp, "Walrus: wireless acoustic location with room-level resolution using ultrasound," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 191–203.

[2] M. Hazas, C. Kray, H. Gellersen, H. Agbota, G. Kortuem, and A. Krohn, "A relative positioning system for co-located mobile devices," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 177–190.

[3] A. Prorok, P. Tomé, and A. Martinoli, "Accommodation of nlos for ultra-wideband tdoa localization in single-and multi-robot systems," in *Indoor Positioning and Indoor Navigation*. IEEE, 2011, pp. 1–9.

[4] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 2000, pp. 32–43.

[5] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 269–280.

[6] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] G. Gudmundsson, B. Jonsson, and L. Amsaleg, "A large-scale performance study of cluster-based high-dimensional indexing," in *Proceedings of the international workshop on Very-large-scale multimedia corpus, mining and retrieval*. ACM, 2010, pp. 31–36.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] P. Bahl and V. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *IEEE INFOCOM*, 2000.

[11] M. Youssef and A. Agrawala, "The horus location determination system," *Wireless Networks*, vol. 14, no. 3, pp. 357–374, 2008.

[12] A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, and V. Otsason, "Gsm indoor localization," *Pervasive and Mobile Computing*, vol. 3, no. 6, pp. 698–720, 2007.

[13] J. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie, "Growing an organic indoor location system," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 271–284.

[14] K. Chintalapudi, A. Padmanabha Iyer, and V. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 173–184.

[15] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "Beepbeep: a high accuracy acoustic ranging system using cots mobile devices," in *Sensys 2007*, 2007.

[16] J. Qiu, D. Chu, X. Meng, and T. Moscibroda, "On the feasibility of real-time phone-to-phone 3d localization," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2011, pp. 190–203.

[17] 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 the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 305–316.

[18] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner, "Virtual compass: relative positioning to sense mobile social interactions," in *Proceedings of the 8th international conference on Pervasive Computing*. Springer-Verlag, 2010, pp. 1–21.

[19] L. K. Torsten Sattler, Bastian Leibe, "Fast image-based localization using direct 2d-to-3d matching," in *International Conference on Computer Vision*. IEEE, 2011, pp. 667–674.

[20] J. Z. Liang, N. Corso, E. Turner, and A. Zakhor, "Image based localization in indoor environments," in *International Conference on Computing for Geospatial Research and Application*. IEEE, 2013, pp. 70–75.

[21] J. Kosecka, L. Zhou, P. Barber, and Z. Duric, "Qualitative image based localization in indoors environments," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2. IEEE, 2003, pp. II–3.

[22] I. Ulrich and I. Nourbakhsh, "Appearance-based place recognition for topological localization," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. Ieee, 2000, pp. 1023–1029.

[23] Y. Tian, R. Gao, K. Bian, F. Ye, T. Wang, Y. Wang, and X. Li, "Towards ubiquitous indoor localization service leveraging environmental physical features," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 55–63.

[24] J. G. Manweiler, P. Jain, and R. Roy Choudhury, "Satellites in our pockets: an object positioning system using smartphones," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 211–224.