# SISE: Self-updating of Indoor Semantic Floorplans for General Entities

Xiaoqiang Teng, *Student Member, IEEE,* Deke Guo, Yulan Guo, Xiang Zhao, and Zhong Liu, *Member, IEEE*

**Abstract**—Indoor semantic floorplan is important for a range of location based service (LBS) applications, attracting many research efforts in several years. In many cases, the out-of-date indoor semantic floorplans would gradually deteriorate and even break down the LBS performance. Thus, it is important to automatically update changed semantics of indoor floorplans caused by environmental variation. However, few research has been focused on the continuous semantic updating problem. This paper presents SISE as a mobile crowdsourcing system that uses a new abstraction for indoor general entities and their semantics, enGraph, to automatically update changed semantics of indoor floorplans using images and inertial data. We first propose efficient methods to generate enGraph. Thus, an image can be associated with an indoor semantic floorplan. Accordingly, we formulate the enGraph matching problem and then propose a quality-based maximum common subgraph matching algorithm so that entities extracted from an image can be corresponded to entities in the indoor semantic floorplan. Furthermore, we propose a quadrant comparison algorithm and a region shrink based localization algorithm to detect and localize changed entities. Thus, the new semantics can be labeled and out-of-date semantics can be removed. Extensive experiments have been conducted on real and synthetic data. Experimental results show that $80\%$ of out-of-date semantics of indoor general entities can be updated by SISE.

**Index Terms**—Indoor semantic floorplan, crowdsourcing, self-updating system

## 1 Introduction

Indoor location-based services (LBS) have been extensively studied in the mobile computing community over the past few years with a variety of applications, including navigation, geo-social networks, and advertisements. Indoor map building has attracted a lot of research attention from both industry and academia. For example, Google Indoor Maps[1] can provide detailed indoor floorplans covering several large public indoor spaces, such as malls, airports, and sport venues. However, only a small fraction of millions of indoor environments can be provided by Google because these indoor floorplans are usually built manually and hence are labor-intensive and time-consuming. Recently, many solutions have been proposed to achieve automatic construction of indoor floorplans using motion trajectories of mobile users [1], [2], [3], [4], [5], [6]. Despite of these progresses, none of those approaches have provided rich semantics associated with indoor floorplans. If the indoor semantic floorplan is available, existing LBSs can be enhanced and new LBSs can be further designed.

Semantics tagged on indoor floorplans represent the attributes (e.g., labels, spatial locations, sizes, and functionalities) of indoor objects in an indoor space. Semantics can be classified as place's semantics (e.g., the location and ID of a room) and entity's semantics (e.g., the location and name of a fridge). Recently, many approaches have been proposed to label or infer semantics for some places and entities in indoor space [7], [8], [9], [10], [11], [12], [13]. First, to label places, ShopProfiler [7], SemSense [8], and AutoLabel [9] were proposed to identify stores in a shopping mall and label their names in indoor floorplans. Second, to label general entities, several Structure from Motion (SfM) based methods [10], [11] were proposed to reconstruct the layout of an indoor space and to label the name of entities in the layout. OverLay [12] was presented to combine multiple techniques to register objects into an augmented reality system. TransitLabel [13] was developed to recognize user activities in transit stations and to infer the functionalities around the physical areas of users.

However, these approaches mainly focus on identifying and labeling semantics of some places and entities. Moreover, even if the indoor semantic floorplan is constructed, the semantics in a dynamic indoor environment may be frequently changed. For example, a furniture in a room might be moved. Consequently, the initial indoor semantic floorplan would gradually deteriorate and even break down the performance of LBSs, if new semantics cannot be labeled or out-of-date semantics cannot be removed. However, research on self-updating of semantics in dynamic indoor environments is very limited.

Recently, mobile crowdsourcing has become a popular concept. It is a low-cost and efficient way to obtain data from mobile users. A large body of LBSs have been developed using the mobile crowdsourcing manner, such as indoor localization [14], [15], indoor navigation [16], [17], and indoor mapping [1], [2], [3], [4], [5], [6], [13], [18]. Besides, the wide availability of mobile devices and wearable devices are

- *X. Teng, D. Guo, X. Zhao, and Z. Liu are with the College of Information System and Management, National University of Defense Technology, Changsha, Hunan,* 410073*, P. R. China.*
- *Y. Guo is with the College of Electronic Science and Engineering, National University of Defense Technology, Changsha, Hunan,* 410073*, P. R. China. Y . Guo is also with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, P. R. China.*
- *E-mail: {tengxiaoqiang13, dekeguo, yulan.guo, xiangzhao, liuzhong} @nudt.edu.cn.*

1. http://www.maps.google.com/.

equipped with built-in visual and inertial sensors. Therefore, mobile users are enabled to take and share geo-tagged pictures anywhere in indoor environments.

Following this trend, we propose a mobile crowdsourcing system (called SISE) to automatically and continuously update semantics of general entities in dynamic indoor environments. We use the ubiquitous sensors available in commodity mobile devices to develop a new abstraction data model, called enGraph, for the representation of indoor general entities and their semantics, which is called enGraph. An enGraph is an undirected graph. If indoor general entities are changed, those enGraphs generated for entities can be different. Therefore, enGraphs provide a method to update changed entities. SISE applies the entity recognition algorithms [19], [20] to recognize different entities in images and uses landmarks to localize the changed entities. Starting from an out-of-date indoor semantic floorplan, SISE can timely obtain a new indoor semantic floorplan to persistently maintain the quality of indoor LBSs for a long term.

Implementing this basic idea into a deployable system, however, faces several challenges. **First**, it is difficult to efficiently generate enGraphs to accurately and completely represent general indoor entities and their semantics. To address this problem, we use multiple entity recognition algorithms to recognize entities in images and their names are an enGraph. An enGraph is then constructed as a complete graph, where a vertex represents an entity, an edge is linked between any two entity vertices, and each vertex has a label set to represent its semantics. Accordingly, we propose two efficient enGraph generation methods to build the data enGraph and the query enGraph. The data enGraph is generated by the indoor semantic floorplan and the query enGraph is generated from an input image. We also propose a coordinate transformation method to align the coordinate systems of these two enGraphs.

**Second**, since semantics extracted from an image are new in an indoor space, it is challenging to correspond entities in an image with the ones in the indoor semantic floorplan to find the out-of-date semantics. A simple initialization method is to match the query enGraph with the data enGraph using an existing subgraph matching method. Existing subgraph matching methods, however, fail to provide an high-quality matching result for our enGraphs in most cases. Those incorrect and low-quality subgraphs may lead to wrong correspondences. To address this problem, we first formulate the enGraph matching problem as a quality-based maximum common subgraph matching problem with set similarity in enGraph. We then incorporate the backtracking, search space shrinking, and the quality score based methods to accurately and efficiently find the high-quality matched subgraph.

**Third**, it is challenging to detect and localize changed entities and to update their semantics at correct positions in an indoor semantic floorplan. To address this problem, we first present a quadrant model to represent general entities and then propose a quadrant comparison based algorithm to detect changed entities in indoor space. To localize changed entities, we then propose a region shrink based localization algorithm under two constraints, i.e., the quadrant constraint among entities and the orientation constraint between the entity and the camera. Thus, SISE can automatically update the out-of-date semantics using the new ones.

The contribution of this work can be summarized as follows. **First**, we formulate the indoor semantic floorplan updating problem for entities and then propose a mobile crowdsourcing based floorplan self-updating method. **Second**, we develop a new abstraction representation for indoor general entities and their semantics (called enGraph) and further propose two efficient enGraph generation methods. The novel abstraction representation provides a method to accurately and efficiently update out-of-date semantics in the indoor semantic floorplan. **Third**, to obtain high-quality matched subgraphs between two enGraphs with set similarity, we design a quality-based subgraph matching algorithm, which greatly improves the semantics updating accuracy of SISE. **Fourth**, we propose a light-weight algorithm to accurately and timely detect and localize changed entities for semantics updating. **Finally**, a SISE prototype system is developed and extensive experiments have been conducted on real and synthetic data. Experimental results show that $80\%$ of out-of-date semantics can be updated. These experimental results have demonstrated the effectiveness of SISE.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 gives an overview of SISE. Section 4 introduces the abstraction representation of indoor general entities and their semantics (called enGraph) and enGraph generation methods. Section 5 presents the quality-based subgraph matching method. Section 6 shows the changed semantics detection and localization algorithms. The prototype implementation and experiments are discussed in Section 7. Technical discussions and limitations are given in Section 8. We conclude the work in Section 9.

## 2 Related Work

Although a large body of literatures are available in LBS, we mainly review those ones closely related to our work.

**Automatic Indoor Floorplan Reconstruction.** Most indoor floorplan construction approaches are based on pedestrian motion traces [1], [2], [3], [4], [5], [6]. CrowdInside [1] was proposed to estimate accurate indoor motion traces of users using anchor points (e.g., elevators) to construct the building's layout. Walkie-Markie [3] and the work in [2] were introduced to combine WiFi signals and inertial data to obtain hallways' length, orientation, and the indoor floorplan. MapGENIE [5] was presented to construct indoor floorplan using exterior information and the grammar which encodes structural information of a building. Jigsaw [4] and CrowdMap [6] were proposed to use a computer vision approach to process crowdsourced images and videos and combine users' motion traces to estimate the building's layout. However, those systems do not provide any semantic information for indoor floorplans.

**Indoor Semantic Floorplan Construction.** Several approaches have been proposed to label semantics in indoor floorplans. SfM technique was proposed in [10], [11], [21] to identify and localize all the foreground objects in a room to construct an indoor semantic floorplan. However,

those SfM approaches do not address the automatical and continuous semantics updating problem. ShopProfiler [7] was proposed to use indoor motion traces of users and WiFi heat map to construct indoor floorplan, identify shops and infer brand names. AutoLabel [9] was proposed to combine website information and WiFi tagged pictures of stores to label store names in the floorplans of malls. SemSense [8] requires users to actively assign a store name to a physical location during check-in operations. TransitLabel [13] was developed to recognize user activities in transit stations and to infer the functionalities around the physical areas of users. However, those techniques mainly focus on the labeling of specific semantics (e.g., store names, transit stations semantics). OverLay [12] combines the smartphone sensing and the computer vision techniques to register objects into an augmented reality system by collecting a set of images. However, it has to manually label all images.

Although a new indoor semantic floorplan can be constructed for every particular time period using those methods, this updating strategy, however, is very labor-intensive and time-consuming, and may introduce unnecessary updates for unchanged environment. Those methods are designed for automatic construction of complete floorplans rather than continuous semantic floorplan updating in complex indoor space. Moreover, these approaches mainly focus on the identification and labeling of semantics of some particular places and entities, such as store names and some semantics of transit stations. For SISE, entity recognition algorithms [19], [20] are used to automatically and continuously recognize general entities from images, detect changed entities and update their semantics in an indoor floorplan. SISE goes one step further and maintains the semantics of entities in an indoor space.

**Subgraph Matching.** Subgraph matching is the basis for many graph applications [22], [23], [24]. Since subgraph matching is NP-hard [25], a large number of methods has been proposed to solve this problem. For exact subgraph matching, the VF2 algorithm [26], TreePi [27], and RINQ [23] have been proposed. However, in our work, the label on each vertex in the query graph is given in probability, which is determined by set similarity. Those exact subgraph matching approaches do not consider set similarity on vertices. For inexact subgraph matching, some vertices or edges can be matched not exactly. Closure-Tree [28] was proposed to support both subgraph query and graph similarity query. SAGA [29] was proposed to search subgraphs in a database that are similar to the query, allowing for node mismatches, node gaps, and graph structural differences. TALE [22] was proposed for approximate subgraph matching to allow node mismatches, node/edge insertion and deletion. Several maximum common subgraph matching methods have been proposed, as reviewed in [30]. However, our work is different from these inexact subgraph matching solutions. That is, no node/edge mismatches are allowed in our work, and the matching vertices should have similar sets.

The work closest to ours was recently proposed in [24]. It used effective pruning and indexing methods to address the subgraph matching problem in a large graph database by set similarity. This work assumes that the data graph is a probability graph while the query graph is a certain graph. However, in our work, the data graph is a certain graph while the query graph is a probability graph. Moreover, the query graph may not be included in the data graph. Therefore, existing subgraph matching solutions cannot be directly applied in our work.

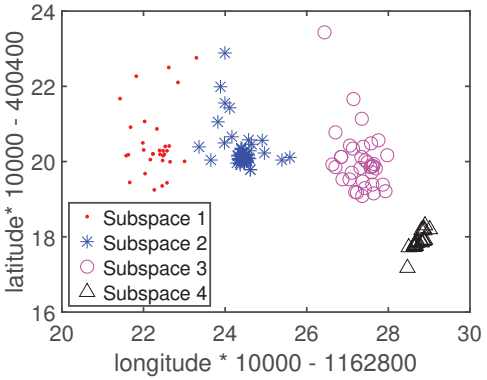



Fig. 1: An illustration of the indoor localization SDK of different subspaces.

## 3 Overview

In this section, we give an overview of this work, including the indoor spatial model, the problem description, and the architecture of the proposed system.

### 3.1 Indoor Spatial Model

The indoor spatial model of a 2D indoor space is built by considering two coordinate systems. The first one is the earth-centred earth-fixed coordinate system (ECEF), which is the world coordinate system. The second one is the floorplan coordinate system (FPCS), which is a local coordinate system. We assume that the updating of each indoor semantic floorplan is independent. For simplicity, the origins of ECEF and FPCS on each floor are assumed to be coincident. Besides, FPCS ($X_{FPCS}$) can be transformed to ECEF ($X_{ECEF}$) using the following equation:

$$X_{ECEF}=R(\omega)X_{FPCS}=\begin{bmatrix}\cos\omega & -\sin\omega\\ \sin\omega & \cos\omega\end{bmatrix}X_{FPCS}, \quad (1)$$

where $\omega$ is the rotation angle and $R(\omega)$ is the rotation matrix between ECEF and FPCS.

Additionally, the indoor space is further divided into a series of subspaces using their own special features: (1) Corridor: both sides of a corridor are walls, or one side is wall and the other side is transparent windows. (2) Room: it contains a portion of space separated by walls. (3) Open Area: it is like a room without walls (e.g., a lobby). To obtain indoor subspaces in an indoor floorplan, the outline of subspaces in an indoor floorplan is manually drew. The size of each subspace is then calculated using an image binaryzation method [31] and the size of the indoor floorplan. The indoor spatial model is suitable for most buildings in the world. In this paper, the existing indoor localization SDK, such as the Baidu LBS SDK[2], is used to locate a user in the indoor space. Specifically, the Baidu LBS SDK uses WiFi, magnetic data, and cell-tower signals to locate a user in an

2. http://lbsyun.baidu.com/location/

indoor space. Although it has an average localization error of few meters, it still can accurately localize the subspace of a user. As shown in Fig. 1, the location points of subspaces 1, 2, 3, and 4 are obviously separated from each other.

### 3.2 Problem Description

An indoor semantic floorplan contains several semantics (e.g., names and positions) extracted from general entities in indoor space. Since indoor environments are usually dynamics, these semantics in the initial indoor semantic floorplan would be out-of-date, resulting in a deteriorated and even break down performance of LBS systems. Therefore, new semantics should be labeled and out-of-date semantics should be removed. This general problem is formalized as *indoor semantic floorplan updating*.

In this paper, our objective is to obtain new semantics of these changed entities and to automatically update the old ones. Specifically, this paper focuses on semantic self-updating of general entities in indoor space. Two semantics of an entity have to be updated, including the category and position $(x, y)$, i.e., $S$=$\{s_i|(category, x, y), 1{\leq}i{\leq}M\}$, where $M$ is the number of indoor entities. Several existing indoor semantic floorplan construction methods can work on crowdsourced images [8], [10], [11], [12]. Same as these previous systems, we also assume that users are motivated to contribute their *images to our system via mobile crowdsensing* [4], [18]. Based on those facts, we aim to design a method to update the indoor semantic floorplan. This method is expected to be efficient while meeting accuracy and confidence requirements.

The *semantic updating accuracy* requirement is defined by two parameters: a confidence interval $\epsilon$ and an error probability $\delta$. Let $\hat{m}$ be the number of updated semantics and $m$ be the number of changed semantics in an indoor space, the updating accuracy is calculated if $\hat{m}$ satisfies $Pr\{|\hat{m}\text{-}m|{\leq}\epsilon m\}{\geq}1\text{-}\delta$. Furthermore, the *localization accuracy* of a changed entity is calculated as the Euclidean distance between the estimated location and the ground-truth location of an entity in an indoor space.

Besides, the self-updating method should be *efficient*. Although existing methods [7], [8], [9], [12], [13] can be used to reconstruct the indoor semantic floorplan for every particular time period, this updating strategy is very labor-intensive and time-consuming.

### 3.3 System Architecture

Fig. 2 illustrates the architecture of the proposed SISE system. SISE is a crowdsourcing-based semantics self-updating system for indoor semantic floorplans. It requires images and inertial data obtained by ordinary mobile devices. It is composed of the following two components. $(I)$ *Mobile application.* It allows user to collect crowdsourced data (images and inertial data), which are then automatically compressed and uploaded to the server for further processing. $(II)$ *Updating engine.* Most of the computational burden is enforced in the updating engine, which consists of three modules, including the enGraph generation module (Section 4), the enGraph matching module (Section 5), and the indoor semantic floorplan updating module (Section 6).

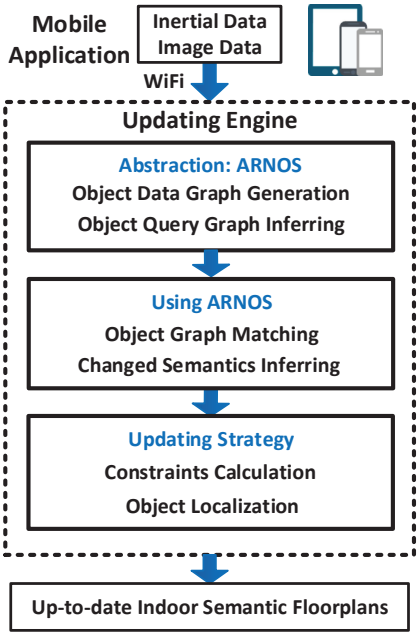



Fig. 2: System architecture.

The updating engine can be implemented on a PC or a cloud server.

**Semantic Detection.** Given an image, entity recognition algorithms have enabled fast and accurate entity recognition using deep learning techniques [19], [20]. We use three entity recognition algorithms, i.e., Faster R-CNN [19], SSD [20], and R-FCN [32]. These methods use neural networks to determine whether an entity is present in an image, and to localize the entity in the image.

**The enGraph Generation Module.** We develop a new abstraction representation for indoor general entities and their semantics, enabling an image to be associated with an indoor semantic floorplan. More specifically, we propose a method for enGraph to generate new data model for an indoor semantic floorplan and an image. We then transform these enGraph local coordinate systems into the world coordinate system.

**The enGraph Matching Module.** To match the data enGraph and the query enGraph, we first formulate the enGraph matching problem and then propose a quality-based maximum common subgraph matching algorithm. Specifically, the quality score of matched subgraphs is calculated to obtain a high-quality result.

**Updating Module.** This module takes the data enGraph, the query enGraph, and their matched subgraph as input to update semantics in the indoor floorplan. Specifically, we propose a quadrant comparison algorithm and a region shrink based localization algorithm to detect and localize changed entities. The new semantics can then be labeled and out-of-date semantics can be removed.

Additionally, since those crowdsourced data may not cover all places of an indoor space, some entities may not be updated. Therefore, similar to other mobile crowdsourcing systems [4], [6], [13], an incentive mechanism [15] can be designed to encourage users to contribute more images.

## 4 enGraph: An Abstraction for Indoor General Entities

We start with a new abstraction data model for indoor general entities and their semantics. We then design two dedicated generation methods for the abstraction data model. Accordingly, we propose a coordinate transformation method to align the coordinate systems of the data model.

### 4.1 A New Abstraction: enGraph

To update out-of-date semantics using images, images have to be associated with the indoor semantic floorplan. To solve this problem, we design a new abstraction data model for entities and their semantics contained in images and the indoor semantic floorplan, called *enGraph*. An enGraph is an undirected graph with each vertex representing an entity and an edge is added between two arbitrary entity vertices. Each vertex has a set of labels to represent semantics of an entity.

**Definition 1** (*Data enGraph*)**.** A *data enGraph* $G$ is represented by a tuple $\langle V(G), E(G), L(G)\rangle$, where $V(G)$ is a set of vertices, $E(G)$ is a set of edges $(E(G)\subseteq V(G)\times V(G))$, and $L(G(u))$ is a label for a vertex $u\in V(G)$.

**Definition 2** (*Query enGraph*)**.** A *query enGraph* $Q$ is defined as $\langle V(Q), E(Q), L(Q), Pr(Q)\rangle$, where $V(Q)$ is a vertex set, $E(Q)$ is an edge set $(E(Q)\subseteq V(Q)\times V(Q))$, $L(Q(v))$ is a label set for a vertex $v\in V(Q)$, and Pr(Q) is a probability set of $L(Q)$.

It can be seen from Definitions 1 and 2 that the data enGraph is a certain graph as it is generated by an entire indoor semantic floorplan. In contrast, the query enGraph is a probabilistic graph as it is generated by an image and the labels on a vertex are given in probability. Besides, the query enGraph may not be included in the data enGraph due to indoor environmental dynamics, such as a newly emerged entity $u$. In this case, $u\notin V(G)$ and $u\in V(Q)$.

Three operations are further defined on two enGraphs ($G_1$ and $G_2$), including intersection, minus and union operations.

**Definition 3** (*intersection operation*)**.** The *intersection operation* of $G_1$ and $G_2$ is to find the largest graph $G_3$ that satisfies $G_3\subseteq G_1$ and $G_3\subseteq G_2$, which is defined as $G_3=G_1\cap G_2$.

**Definition 4** (*minus operation*)**.** Suppose that $G_2\subseteq G_1$, the *minus operation* of $G_1$ and $G_2$ is to find the largest graph $G_3$ that satisfies $G_3\subseteq G_1$ and $G_3\cap G_2=\varnothing$, which is defined as $G_3=G_1/G_2$. If $G_2$ is a vertex set, $G_3$ is a new graph obtained by removing the vertices of $G_1$ that are included in $G_2$.

**Definition 5** (*union operation*)**.** The *union operation* of $G_1$ and $G_2$ is to find a graph $G_3$ by adding $G_1/G_2$ into $G_2$ or $G_2/G_1$ into $G_1$, which is defined as $G_3=G_1\cup G_2$.

### 4.2 The Data Generation Method

To accurately and efficiently detect the changed semantics in an indoor semantic floorplan, the data enGraph should be *efficient* and *simple*. A data enGraph is efficient means it encodes all general entities and their semantics. A data enGraph is simple means it requires a short computational time for graph processing to meet required accuracy. Therefore, it is challenging to construct a useful and simple data enGraph. To solve this problem, the construction roles of the data enGraph are designed:

1) Each subspace is independently used to construct the data enGraph, and these data enGraphs are then connected using the special entities (e.g., doors) which can easily be identified in an indoor semantic floorplan.
2) The data enGraph constructed in each subspace is a complete graph.
3) Equation 1 is used to transform FPCS to ECEF of the data enGraph.

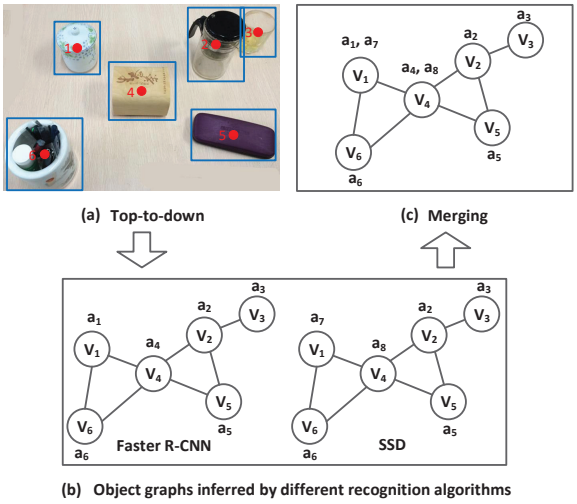



Fig. 3: An illustration of the query enGraph generation method. (a) Entity recognition results $a_i$ obtained by the Faster R-CNN and the SSD algorithms. (b) The object graph is inferred using entity locations in the image coordinate system. (c) The query enGraph is generated by merging these two object graphs.

### 4.3 The Query enGraph Generation Method

In this section, we propose a query enGraph generation method. Here, an example is used to illustrate this method. We assume that six entities are included in an image, as shown in Fig. 3.

First, two object recognition algorithms (i.e., Faster R-CNN [19] and SSD [20]) are used to recognize entities from an image in this example. The details of these two entity recognition algorithms are not presented due to space limit. Their output results include the name $a_i$, the location in the image coordinate system, and the bounding box of each entity, as shown in Fig. 3(a). Here, a bounding box represents the minimum bounding rectangle of an entity in an image.

Second, an object graph is constructed using the location $(u, v)$ of entities in the image coordinate system [33]. In an object graph, a vertex represents an entity, an edge is linked between two vertices, and the label denotes the entity name. As shown in Fig. 3(b), two object graphs are constructed using the Faster R-CNN and the SSD algorithms.

Third, these object graphs are merged to generate the query enGraph using the union operation. Since the labels of

a vertex produced by different entity recognition algorithms may be different (See Fig. 3(b)), a label set is used. As illustrated in Figs. 3(b-c), vertex $V_1$ has two different labels in two graphs, the label set of a vertex $V_1$ is set $\{a_1, a_7\}$.

### 4.4 Coordinate Transformation

As the data enGraph is defined in FPCS and the query enGraph is defined in the camera coordinate system, the goal of coordinate transformation is to transform FPCS and the camera coordinate system into ECEF.

First, Eq. 1 is used to transform FPCS into ECEF for the data enGraph. Second, to transform the camera coordinate system into ECEF for the query enGraph, we propose a coordinate system transformation method. Given an enGraph $Q^C$ in the camera coordinate system and an enGraph $Q^E$ in ECEF, we use two matrixes, i.e., translation matrix ($T$) and rotation matrix ($R$), to achieve $Q^E$=$[T, R]Q^C$. For simplicity, the translation between $Q^E$ and $Q^C$ is not considered in this paper, because the location of objects in the indoor space is currently unknown. Therefore, we have $Q^E$=$RQ^C$.

**Theorem 1.** *Given two enGraphs $Q^{v1}$ and $Q^{v2}$ and the rotation matrix of the camera $R^c$, we can obtain $Q^{v1}$=$R^cQ^{v2}$. Note that, $Q^{v1}$ and $Q^{v2}$ are inferred from two images captured from two viewing directions by the same camera.*

*Proof.* Theorem 1 can be proved by the homography theory in computer vision. Given several entities $O$=$\{o_i|1\leq i\leq m\}$ on the same plane, two intrinsic matrixes $K$ and $K'$ of two cameras, and the rotation matrix $R^c$ between two cameras, we can obtain the homography matrix $H$ as $H$=$K'R^cK^{-1}$ using the homography theory [34]. $m$ is the number of detected entities from an image. For each entity, we have $o_i^{v1}$=$Ho_i^{v2}$=$K'R^cK^{-1}o_i^{v2}$, where $i$=$1, 2, \ldots, m$. As $o_i^{v1}\in Q^{v1}$ and $o_i^{v2}\in Q^{v2}$, thus we have $Q^{v1}$=$R^cQ^{v2}$. Hence, Theorem 1 is proved. □

In our work, the inertial data are used to calculate the attitude of the first camera using the method proposed in [35]. The attitude of the second camera is set in ECEF to make the viewing direction parallels to the $Y_e$ axis.

## 5 enGraph Matching

The data enGraph represents the indoor semantic floorplan and the query enGraph represents the semantics abstracted from an images. It is challenging to accurately match the query enGraph with the data enGraph so that entities extracted from an image can be corresponded to the entities in the indoor semantic floorplan. In this section, we first describe the enGraph matching problem and then present our subgraph matching algorithm to solve this problem.

### 5.1 Problem Description

In this subsection, we first introduce the definition of subgraph isomorphism and then formally describe the subgraph matching problem, namely quality-based maximum common subgraph matching with set similarity in enGraph. Specifically, given a data enGraph $G$ and a query enGraph $Q$, the maximum common subgraph $C$ between $G$ and $Q$ is taken by $C$=$G\cap Q$ with the maximum number of vertices $|C|$.

**Definition 6** (*Subgraph Isomorphism*)**.** Given two graphs $G$=$\langle V(G), E(G), L(G)\rangle$ and $Q$=$\langle V(Q), E(Q), L(Q)\rangle$, a *subgraph isomorphism* from $G$ and $Q$ is an injective function $f: V(G)\rightarrow V(Q)$, such that $\forall(u, v)\in E(G)$, $(f(u), f(v))\in E(Q)$ and $L(G(u))$=$f(L(Q(u)))$, $L(G(v))$=$f(L(Q(v)))$, and $L(G(u, v))$=$f(L(Q(u, v)))$.

**Definition 7** (*Quality-based Maximum Common Subgraph Matching with Set Similarity in enGraph*)**.** For a data enGraph $G$ and a query enGraph $Q$ with $n$ vertices $(v_1, \ldots, v_n)$, and a predefined similarity threshold $\kappa$, a maximum common subgraph match of $Q$ is a graph $C$ of $G$ containing $|C|$ vertices $(v_1, \ldots, v_{|C|})$ of $V(G)$ that satisfies the following conditions:

1) $Q_C$ is isomorphic to $G_C$, where $G_C$ and $Q_C$ represent the maximum common subgraph $C$ in enGraphs $G$ and $Q$.
2) $sim(L(u_i), L(v_j))\geq\kappa$, where $L(u_i)$ and $L(v_j)$ are the sets associated with $u_i$ and $v_j$ in $G_C$ and $Q_C$, respectively. $sim(L(u_i), L(v_j))$ gives a similarity score between $L(u_i)$ and $L(v_j)$.
3) The matched subgraph with the highest quality score determines the final result.

By a reduction from the well-known subgraph matching problem, the quality-based maximum common subgraph matching with set similarity in enGraph can be proved to be NP-complete [25]. In this paper, the *probability Jaccard similarity* is used.

**Definition 8** (*Probability Jaccard Similarity*)**.** Given label sets $L(u)$ and $L(v)$ of vertices $u$ and $v$, the probability Jaccard similarity between $L(u)$ and $L(v)$ is:

$$sim(L(u), L(v))=\frac{\sum_{l\in L(u)\cap L(v)} Pr(l)}{\sum_{l\in L(u)\cup L(v)} Pr(l)}, \quad (2)$$

where $Pr(l)$ is the probability of label $l$, $Pr(l)\geq 0$.

Furthermore, the quality score is used to choose the best result. In our work, two matrices, the distance matrix and probability matrix, are used to compute the quality score. The distance matrix $D$ is the Euclidean distance between the matched subgraph and the camera, where the location of camera is estimated using the state-of-the-art dead reckoning method [14], [36]. The probability matrix is calculated using the probability Jaccard similarities for labels on vertices in the matched subgraph. Therefore, the quality score $score(C)$ is defined as:

$$score(C)=\frac{1}{Z}(\frac{1}{D}+\sum_{u\in G, v\in Q} sim(L(u), L(v))), \quad (3)$$

where $Z$ is the normalized term.

### 5.2 The MAS Algorithm

In this subsection, we propose a quality-based maximum common subgraph matching algorithm (called MAS) to solve the matching problem between two enGraphs. Without loss of generality, we assume that $G$ and $Q$ are connected and simple graphs, where a simple graph is a graph without self-loops nor multiple edges. Our MAS algorithm can be easily extended to directed and/or edge labeled graphs.

**Algorithm 1** The MAS Algorithm

**Input:** A data enGraph $G'$, a query enGraph $Q$, subspace constraint, orientation constraint, and a predefined similarity threshold $\kappa$.
**Output:** A matched maximum common subgraph $C$.

1: **for** $v \in V(Q)$ **do**
2: **if** $sim(L(v), L(u)) \geq \kappa$ **then**
3: $Z(v) \leftarrow \{u | u \in V(G')\}$;
4: **else**
5: $E_C.add(v)$
6: $Q'$=$Q(V(Q)/E_C)$;
7: **for** $v \in V(Q')$ **do**
8: $Recursive\_Search(v_i)$
9: $g_c \leftarrow G^C.size$
10: **for** $j$=0 to $g_c$-1 **do**
11: $S_c(j)$=$score(G_C(j))$;
12: $C$= $\underset{g}{\text{argmax}}$ $S_c$;
13: **return** $C$;
14: **Function** $Recursive\_Search(v_i)$
15: **for** $u \in Z(v_i)$ and $u$ is unmatched **do**
16: **if** $sim(L(u), L(v_i)) \geq \kappa$ **then**
17: continue;
18: $f(v_i) \leftarrow u$; $u \leftarrow matched$;
19: **if** $i$<$|V(Q)|$-$E_C.size$ **then**
20: $Recursive\_Search(v_{i+1})$;
21: **else**
22: $G_C.add(f(Q'))$;
23: $f(v_i) \leftarrow NULL$; $u \leftarrow unmatched$;

The MAS algorithm is implemented as a backtracking algorithm [37]. The idea of the backtracking algorithm is to find solutions by increasing or abandoning partial solutions. Instead of searching over the whole space of a data enGraph $G$, we limit the search space by enforcing two constraints using the indoor spatial model and the quadrant model. (1) Subspace constraint. The search space can be easily limited to a special subspace based on the location of a user, such as a room. (2) Orientation constraint. The orientation of a user for image acquisition can be estimated using the inertial data [35]. Therefore, the searching space is limited in a small data enGraph $G'$ using these constraints such that the subgraph matching process can be accelerated.

Algorithm 1 shows the details of our MAS algorithm. Its inputs include a query enGraph $Q$, a data enGraph $G'$, and a predefined similarity threshold $\kappa$. Its outputs include a matched maximum common subgraph $C$ in $G'$ and a vertex set $E_C$, where each vertex in $Q$ is not included in $G'$. First, a matching candidate set $Z(v)$ is found for each vertex in $Q$ (Lines 2-3). At the same time, the $E_C$ is also found (Lines 4-5) using the probability Jaccard similarity between vertices $u$ and $v$, where $u \in G'$ and $v \in Q$. Once the candidate set is obtained, the new query enGraph $Q'$ is generated by removing these vertices included in $E_C$ (Line 6). Next, the function $Recursive\_Search(v_i)$ is used to match $v_i$ with $Z(v_i)$ (Lines 7-8). This process is repeated by recursively matching the subsequent vertex $v_{i+1}$ with $Z(v_{i+1})$ (Lines 19-20). If every vertex of $Q'$ has a counterpart in $G'$ (Line 22). If all vertices in $Z(v_i)$ have been tested and no feasible matching can be found, $Recursive\_Search(v_i)$ is backtracked to the previous state for further exploration (Line 23). If there are multiple matched subgraphs in $G'$, the quality score is calculated for each matched subgraph (Line 11). The subgraph with the highest quality score determines the final result.

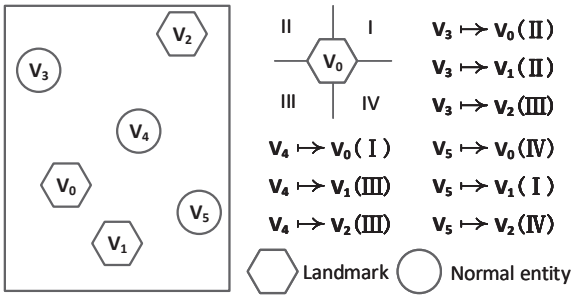



Fig. 4: An illustration for the quadrant model of entities. Entities $V_0$, $V_1$, and $V2$ are landmarks and entities $V_3$, $V_4$, and $V_5$ are normal entities. Their location relationships can be obtained.

**Theorem 2.** *The time complexity of the MAS algorithm is* $O(|V(Q)| \times |V(G')| + |V(Q')| \times |Z(v)| + g_c)$.

*Proof.* Algorithm 1 consists of three stages, i.e., matching candidate searching, $Recursive\_Search(v_i)$, and quality score calculation. In the first stage, the time complexity for matching candidate searching is $O(|V(Q)| \times |V(G')|)$ since it processes $|V(Q)|$ vertices in graph $Q$ and $|V(G')|$ vertices in graph $G'$. In the second stage, the time complexity for traversing all vertices in $Q'$ and $Z(v)$ is $O(|V(Q')| \times |Z(v)|)$ in the worst case. The time complexity for quality score calculation over all matched subgraph in $G'$ is $O(g_c)$. Thus, the overall complexity is $O(|V(Q)| \times |V(G')| + |V(Q')| \times |Z(v)| + g_c)$. Hence, Theorem 2 is proved. □

## 6 Indoor Semantic Floorplan Updating

If all entities extracted from images have been corresponded to the indoor semantic floorplan, we are able to update changed semantics in an indoor semantic floorplan. A critical issue is to detect and localize changed entities and update their semantics. To solve this problem, we first present the quadrant model for general entities and then propose a quadrant comparison based algorithm to detect changed entities in the indoor space. Second, to localize changed entities, we propose a region shrink based localization algorithm using two constraints, the quadrant constraint among entities and the orientation constraint between the entity and the camera.

### 6.1 Quadrant Model for General Entities

The quadrant model is built on an entity and has a local coordinate system with its axes parallel to the axes of ECEF. Specially, it has $h$ quadrants (I, II, ...). Thus, the location relationship between two entities is described using the quadrant. As illustrated in Fig. 4, the quadrant model of the entity $V_0$ has four quadrants (I, II, III, IV). Take entity $V_0$ for example, entity $V_3$ is on the second quadrant of the entity $V_0$

**Algorithm 2** The Quadrant Comparison Algorithm

**Input:** The matched subgraph $G_C$ in data enGraph, query enGraph $Q$, landmark set $E_L$, and $E_C$ (Algorithm 1).
**Output:** The changed entity set $C_E$.

1: $C_E$=$\emptyset$; $G_C^{'}$=$G_C/E_L$; $Q'$=$Q/E_C$;
2: **for** $v \in G_C$ **do**
3: $db_1$=$ConstructEntityDatabase(G_C(v))$;
4: **for** $u \in Q'$ **do**
5: $db_2$=$ConstructEntityDatabase(Q'(u))$;
6: **for** $v \in G_C^{'}$ **do**
7: **if** $Comparison(db_1(v), db_2(v))$==$true$ **then**
8: $C_E.add(v)$; break;
9: **return** $C_E$;

and their location relationship can be denoted as $V_3 \mapsto V_0$(II). Similarly, the location relationships between other entities ($V_3$-$V_5$) and the entity $V_0$ can be obtained. Here, the entity $V_0$ is denoted as the landmark and entities $V_3$, $V_4$ and $V_5$ are denoted as normal entities. As shown in Fig. 4, there are three landmarks ($V_0$-$V_2$) and three normal entities ($V_3$-$V_5$). Besides, if the location of a normal entity in the indoor space is changed, the entity is considered as a changed entity.

### 6.2 Algorithm for Changed Entity Detection

The goal of changed entity detection is to find these changed entities in a specific indoor area. These changed entities include moved entities, disappeared entities, and newly emerged entities. So far, the newly emerged entities have been obtained from Algorithm 1, that is, these entities are included in $E_C$. Therefore, our goal is to detect the moved and disappeared entities. There are three intuitions behind changed entity detection. (1) There are many fixed entities in indoor space, such as doors, windows, wardrobes, and elevators. Those entities are considered as *landmarks*. If those fixed entities are changed, it is very hard to select landmarks without prior knowledge. In the worst case, human intervention is still required by our system to select landmarks. (2) The location of other entities can be estimated using these landmarks and are further considered as new landmarks. (3) If the location of an entity changes, the quadrant relationship between the changed entity and landmarks will be changed. Therefore, a quadrant comparison based algorithm is proposed using these intuitions. The basic idea of the quadrant comparison based algorithm is to compare quadrants between the normal entity and landmarks. First, entity databases are constructed to store landmarks, normal entities, and the quadrants for these entities included in the data enGraph and the query enGraph. If quadrants between a normal entity and landmarks are inconsistent in the two entity databases, the normal entity is considered as a changed entity.

Algorithm 2 illustrates the details of the quadrant comparison algorithm. First, the normal entity set is initialized to be $\emptyset$, landmarks are removed from the matched subgraph $G_C$ in the data enGraph, and the entities included in $E_C$ are also removed from the query enGraph $Q$ (Line 1). Then, two entity databases are constructed using enGraphs $G_C$ and $Q$ (Lines 2 to 5). If quadrants between a normal entity and landmarks are changed, the normal entity is then considered as a changed entity and is added to the changed entity set $C_E$ (Lines 6 to 8).

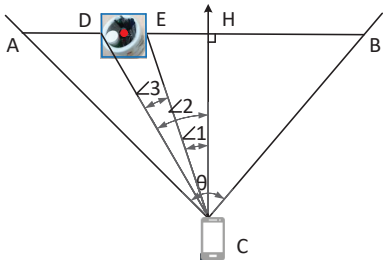



Fig. 5: An example of the orientation constraint between an entity and a camera.

**Algorithm Analysis.** In Algorithm 2, a changed entity may not be identified if it moves in the same quadrant determined by all landmarks. The probability of this event occurrence is defined as the false negative rate. In fact, if the number of landmarks and quadrants is increased, the false negative rate will be decreased. Therefore, the number of quadrants and landmarks plays a very important role for the detection of changed entities. Theoretical analysis for false negative rate produced by Algorithm 2 is further presented.

Let $q$ be the number of quadrants in the quadrant model of an entity and $r$ be the number of landmarks. For a landmark, the false negative rate $Pr(q, r)$ of a normal entity is calculated as $\frac{1}{q}$. It is the same for multiple landmarks. First, we consider the worse case, that is, there are two quadrants ($q$=2) in the quadrant model of an entity and multiple landmarks are collinear. In this case, the false negative rate is calculate as $\frac{1}{2}$. Therefore, if there are $q$ quadrants in the quadrant model of an entity, the maximum of the false negative rate is $\frac{1}{q}$. Second, we consider the best case that if there are $q$ quadrants in the quadrant model of an entity and all of the $r$ landmarks are not collinear, the minimum false negative rate is $\frac{1}{q^r}$. Assume that the detection for different entities is independent and the false negative rate is therefore within the range of $[\frac{1}{q^r}, \frac{1}{q}]$.

### 6.3 Algorithm for Changed Entity Localization

We start with the calculation of two constraints and then present the changed entity localization algorithm.

**Constraint calculation.** To estimate the location of changed entities, two constraints are calculated using landmarks, including the quadrant constraint among entities and the orientation constraint between the entity and the camera.

*Quadrant Constraint.* The quadrant constraint is determined by the quadrant model (Section 6.1). Any two entities are subject to this specific quadrant constraint. Such quadrant constraint among entities provide the opportunity for the entity localization.

*Orientation Constraint.* Fig. 5 shows an entity with a bounding box (i.e., the blue box), the known location and pose of the camera $C$ in ECEF. The pose of a camera is calculated in the earth coordinate system using magnetometer and gyroscope sensors. The magnetometer sensor provides the absolute direction in the earth coordinate system

**Algorithm 3** The ResLoc Algorithm

**Input:** The matched subgraph $C$ in data enGraph, changed entity set $C_E$, quadrant constraint set $QST$, and orientation constraint set $OST$.

**Output:** The estimated location of changed entities and the updated enGraph.

1: $G_o$=$C/C_E$;
2: **for** $i$=0 to $|C_E|$ **do**
3: $A_q$=$LocationArea(QST(i))$;
4: $A_o$=$LocationArea(OST(i))$;
5: $Region$=$RegionShrink(A_q, A_o)$;
6: $Location(i)$=$CentroidEstimation(Region)$;
7: $G_o.add(C_{E_i})$;
8: **return** $G_o$;

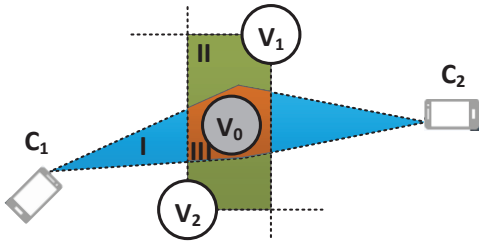



Fig. 6: An example of the ResLoc algorithm. The entity $V_0$ is located in region $A_0$ based on quadrant constraints of other entities ($V_1$ and $V_2$) and in region $A_1$ based on orientation constraints of cameras ($C_1$ and $C_2$). The joint part $A_2$ of these regions is the final region of the entity $V_0$ and the gravity center of region $A_2$ determines the location of entity $V_0$

and the gyroscope sensor provides the relative direction changes with respect to the device platform. However, the pose estimated by magnetometer and gyroscope sensors has errors due to electromagnetic interference. In this paper, the $A^3$ algorithm [35] is used to provide accurate pose. The $A^3$ algorithm primarily leverages the gyroscope, but incorporates the accelerometer and magnetometer to select the best sensing capabilities, resulting in the most accurate attitude estimation [35]. $\theta$ is the camera's Field of View (FoV) and $H$ is the camera's focus, which are fixed for a given mobile device. The orientation angle ($\angle 3$) between the entity and the camera is calculated as follows. First, the lengths of lines $|\vec{HB}|$, $|\vec{EH}|$, and $|\vec{DH}|$ are calculated using points $H$, $B$, $E$, and $D$ in the image. Then, we can obtain $|\vec{CH}|$=$\frac{|\vec{HB}|}{\tan\frac{\theta}{2}}$. Thus, $\angle 1 = \arctan\frac{|\vec{EH}|\tan\frac{\theta}{2}}{|\vec{HB}|}$. Similarity, $\angle 2 = \arctan\frac{|\vec{DH}|\tan\frac{\theta}{2}}{|\vec{HB}|}$. Finally, $\angle 3$ is calculated as $\angle 3$=$\angle 2$-$\angle 1$.

**Localization.** Combining Algorithm 2 for changed entity detection and two constraints, we propose the Region Shrinking based Localization (ResLoc) algorithm to locate the changed entities and to update their semantics in indoor floorplans. The key insight is that the region calculated by two constraints can automatically be shrunk to a small region. Here, an example is used to illustrate this algorithm.

As illustrated in Fig. 6, let the entity $V_0$ (the grey circle) be a changed entity, entities $V_1$ and $V_2$ (white circles) and cameras $C_1$ and $C_2$ are landmarks in the indoor subspace. We can obtain $V_0 \mapsto V_1$(III) and $V_0 \mapsto V_2$(I) using quadrant constraints of entities $V_1$ and $V_2$. Since the locations of the entities $V_1$ and $V_2$ are available, the location of entity $V_0$ is restricted in region $A_0$ (the blue polygon in Fig. 6). Similarly, the location of entity $V_0$ is restricted in region $A_1$ (the green polygon in Fig. 6) using the orientation constraints of cameras $C_1$ and $C_2$. Furthermore, the joint part of these regions (region $A_2$) is the final region of the entity $V_0$. Finally, the gravity center of region $A_2$ determines the location of entity $V_0$. Note that, as the number of landmarks increases, the localization accuracy can be improved (See Fig. 9(b)). Thus, multiple images are used to estimate the location of general entities. We use the density-based clustering algorithm (DBSCAN [38], [39]) to obtain images of an entity.

The ResLoc algorithm is depicted in Algorithm 3. The inputs of the ResLoc algorithm include the matched subgraph $C$ (obtained from Algorithm 1), the changed entity set $E_C$ (obtained from Section 6.2), the quadrant constraint set $QST$, and the orientation constraint set $OST$. The outputs of the ResLoc algorithm include the estimated location of general entities and the updated enGraph. First, the object graph ($G_o$) is extracted as: $G_o$=$C/C_o$ (Line 1). That is, the changed entities are removed from the matched subgraph $C$. It is the part of the entire data enGraph $G$ that we want to update. Lines 2 to 8 describes the process to estimate the location of each changed entity by shrinking the region determined by each landmark. Specifically, regions of the $i$-th entity are obtained using the quadrant constraint and the orientation constraint, respectively. The region of each entity is updated by extracting the joint part of those regions obtained from Lines 3 to 4. Next, the gravity center is used to estimate the final location of each general entity (Line 6). Finally, the changed entities and their semantics are labelled in the indoor semantic floorplan (Line 7).

## 7 Implementation and Evaluation

In this section, the implementation details of SISE are first presented. The evaluation methodology and setups are then described. The performance of each component of SISE is further presented.

### 7.1 Implementation

A SISE prototype was implemented to update the semantic indoor floorplans. It consists of a mobile application software which collects crowdsourced data from various sensors embedded in mobile devices, and a server for data processing. The mobile application software was implemented in Java for Android platform, and has been tested on different Android mobile devices, including Samsung Galaxy S7, MI3, and Huawei P9. All modules of the updating engine were implemented in Python and Java on a PC server with 32GB RAM, an i7 CPU processor, and a 12GB Titan GPU. To recognize entities, parts of the ImageNet[3] and the COCO[4] datasets containing 152 types of indoor general entities, were used to train models of the entity recognition algorithms. Besides, WiFi networks were used for communication between the mobile devices and the server.

3. www.image-net.org/
4. www.mscoco.org/

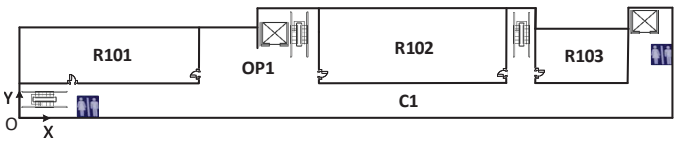



Fig. 7: Experiment scenario for our SISE system, where the area $R$ represents the room, the area $C$ represents the corridor, and the area $OP$ represents the open area.

### 7.2 Evaluation Methodology and Setups

Using real and synthetic data, extensive experiments were conducted to evaluate the performance of each component of SISE. The real data were collected on one floor of our office building by volunteers, which covers $4000\text{m}^2$ with the length of $100$m and the width of $40$m. The floorplan of our office building was further divided into three areas, i.e., three rooms (R101, R102, and R103), a corridor $(C1)$, and an open area $(OP1)$, as shown in Fig. 7. Specifically, rooms R101 and R103 are office rooms and room R102 is a gym. Besides, these areas contain rich information of entities, including their category, location, and size. During the initialization of our experiment, semantics of all entities (e.g., names and locations) were labeled manually since the available semantic floorplans of the building were out of date. Similarly, the ground-truth semantics of changed entities were also labeled manually by recording and watching videos.

Volunteers were invited to participate in the data collection procedure. Each of them carried a mobile device, which is installed with a mobile application software, to capture images at different times of a day. The locations were determined by themselves. These captured images covered most indoor areas and all available general entities. Considering the rewarding mechanism, this data collection method is acceptable for general realistic scenarios. $640$ fully-labeled images were used to test the performance of SISE. Specifically, each subspace was collected $128$ images on average. Those images covered entire area of each subspace. Besides, we also created the synthetic data to test the performance of our method. *@R* denotes that real data are used in the experiment and *@RN* denotes that the real data has $N$ entities. The synthetic data were randomly generated from the same space with $100$ unit length and $40$ unit width. Similarly, *@S* denotes that synthetic data are used in our experiments and *@SN* means that the synthetic data has $N$ entities. For example, *@R*$100$ means that the real data with $100$ entities are used in the experiment.

### 7.3 Performance Evaluation

#### 7.3.1 Performance of Indoor Semantic Floorplan Updating

**Performance Metrics.** *Accuracy* is an important metric for indoor semantic floorplan updating. Suppose that the ground-truth set of changed semantics is $M$ and the set of semantics updated by SISE is $\hat{M}$. In order to assess the performance of SISE, the precision $P$ and recall $R$ of semantic updating are used, i.e.,

$$P=\frac{|\hat{M}\cap M|}{|\hat{M}|}, \tag{4}$$

$$R=\frac{|\hat{M}\cap M|}{|M|}, \tag{5}$$

where $P$ presents the precision of semantic updating, it was used to evaluate the updating accuracy of SISE. $R$ is the recall of semantic updating, it is expressed as the ratio of the number of updated semantics to the number of the groundtruth changed semantics.

Besides, the updating time consumed by changed semantics, is used to evaluate the *efficiency* of SISE. Specifically, the updating time for an image was calculated. We further compared the updating time costed by SISE on all images with that costed by the SFM-based technique.

Furthermore, the entity localization error is another important performance metric to evaluate the performance of SISE. Location error represents the Euclidean distance between the estimated location and the ground-truth location of an entity.

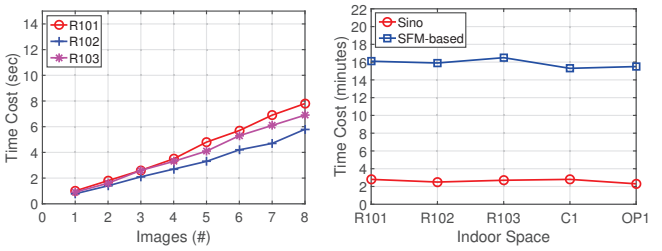
**Experimental Results.** First, the semantic updating performance was evaluated on both real and synthetic datasets in five different indoor subspaces. The Faster R-CNN [19], the SSD [20], and the R-FCN [32] algorithms were used to recognize entities in images. It can be seen from Table 1 that SISE achieves a precision around $81.1\%$ and a recall around $79.8\%$ on real data. It also achieves a precision around $90.5\%$ and a recall around $88.6\%$ on synthetic data. These results clearly demonstrate that our entity semantic updating method is accurate and robust to indoor scenes. The difference between real data and synthetic data are that the real data is generated with the entity recognition algorithms, while the synthetic data does not. In another words, experimental results produced on the synthetic data are only affected by the semantics updating algorithm of SISE. In contrast, the experimental results produced on real data are affected by both the entity recognition algorithms and the semantics updating algorithm. This demonstrates that the performance of entity recognition has a significant influence on the overall performance of SISE. Next, the performance of entity recognition algorithms was further tested using the real data.

In our experiments, three entity recognition algorithms were tested, including the Faster R-CNN [19], the SSD [20], and the R-FCN [32] algorithms. The average recognition precision (mAP) was used to measure the performance of the entity recognition algorithm. It can be seen from Table 2 that if more entity recognition algorithms are used, the mAP value is increased. Experimental results demonstrate that the semantics updating accuracy produced by SISE (Table 1) can further be improved if more entity recognition algorithms are used.

Next, the updating time was measured on real data. The updating time for one entity was first tested using different number of images in different indoor subspaces. The number of images was ranged from $1$ to $8$. The numbers of entities were $256$, $32$, and $128$ for room R101, R102,

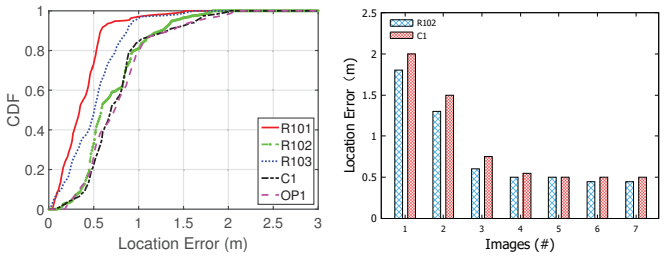
TABLE 1: Update Performance Evaluation on Real Data and Synthetic Data.

| | Precesion ($P$) | Recall ($R$) |
|---|---|---|
| Real data | 81.1% | 79.8% |
| Synthetic data | 90.5% | 88.6% |




(a) Time cost for one entity (b) Time cost comparison

Fig. 8: Time cost for (a) semantic updating of an entity by SISE using different numbers of images and (b) semantics updating by SISE and SFM-based in different indoor spaces using 10 images.




(a) Localization accuracy (b) Query enGraph

Fig. 9: (a) The location errors produced by SISE in different indoor spaces areas. (b) Effect of the number of images on the accuracy of entity location estimation.

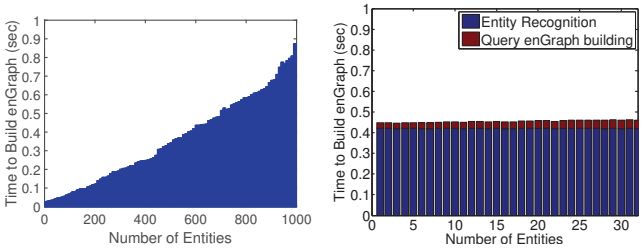
and R103, respectively. Experiments were run 10 times for each indoor subspace. One changed entity was randomly selected in each experiment. Their average results are presented in Fig. 8(a). The time to update the semantic of one entity is less than 9s, which fully demonstrates the efficiency of SISE.

Furthermore, the updating time for all entities of each subspace was tested using 640 images. 128 images were collected in average for each subspace. We extended the SFM technique (namely, COLMAP [40], [41]) to support semantic updating of indoor environments. COLMAP recovers a dense representation of a scene (i.e., the 3D model) using the SFM technique. To obtain the initial $3D$ models, 4800 images were collected in five different indoor subspaces of our office building and five dense 3D models were generated. In our experiments, the number of images used by COLMAP is the same as our method in the same scene. All parameter settings for COLMAP were consistent with the settings presented in the original papers [40], [41]. We manually labeled the differences, i.e., changed entities and their semantics, between two 3D models generated at different time periods of the same scene. Average time consumption was calculated by updating the semantics of each scene. It can be seen from Fig. 8(b) that the time costed by SISE to update the semantics of all entities is 2.7 minutes, while the SFM-based method takes 15.8 minutes in average excluding manual semantic labeling. The experimental results have fully demonstrated the significant advantage of our proposed method over the SFM-based method. Our method is highly efficient due to the use of efficient enGraph data structure. Besides, our method can perform self-updating while the SFM-based method requires manual labeling.

Finally, the entity location localization was further tested in our experiments on real data. The changed entities were randomly selected and the number of changed entities was ranged from 50 to 100 in five different indoor subspaces.

TABLE 2: Recognition Results Produced by Three Entity Recognition Algorithms

| | mAP |
|---|---|
| Faster R-CNN | 44.6 |
| Faster R-CNN + SSD | 58.5 |
| Faster R-CNN + SSD + R-FCN | 79.8 |




(a) Data enGraph (b) Query enGraph

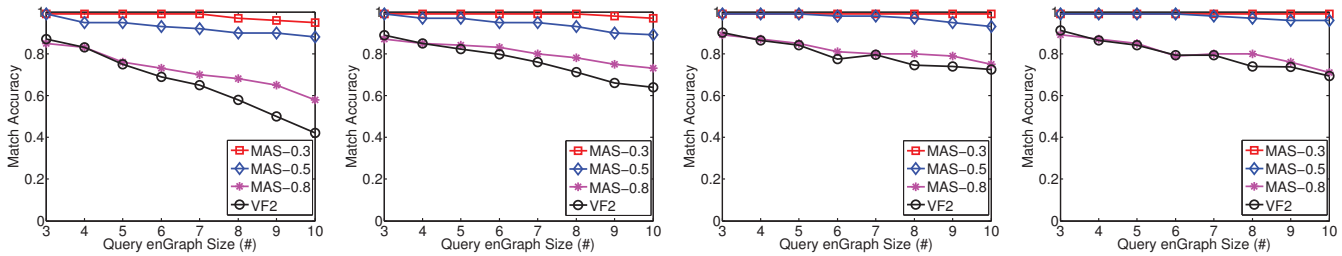
Fig. 10: Time required to generate enGraphs for the data enGraph and the query enGraph.

The ground-truth location of a changed entity was recorded manually. It can be seen from Fig. 9(a) that 90% of changed entities have a location error less than 1.5m. In particular, 90% of entities in Rooms R101 and R102 have a location error less than 0.85m. The largest location error in Corridor $C1$ is close to 2m. Besides, the location errors achieved in Rooms R101 and R103 are smaller than that of other subspaces. Both the number and density of entities in Rooms R101 and R103 are larger than those of other subspaces.

To decrease the location error, multiple images were used. On one hand, multiple images increase the number of entity landmarks. On the other hand, multiple images also provide additional camera constraints. To test the entity localization performance using multiple images, the location errors achieved in corridor $C1$ and room R102 were measured. The number of images are ranged from 1 to 8 with a step of 1. Each experiment was run 10 times. Fig. 9(b) reports their average experimental results. The location error is decreased when the number of images is increased. However, the location error cannot be further reduced when the number of images is close to 5. That is because the instantaneous location and orientation errors of the camera produced by the dead reckoning method [14], [36] and the $A^3$ algorithm [35] still influence the location estimation of normal entities.

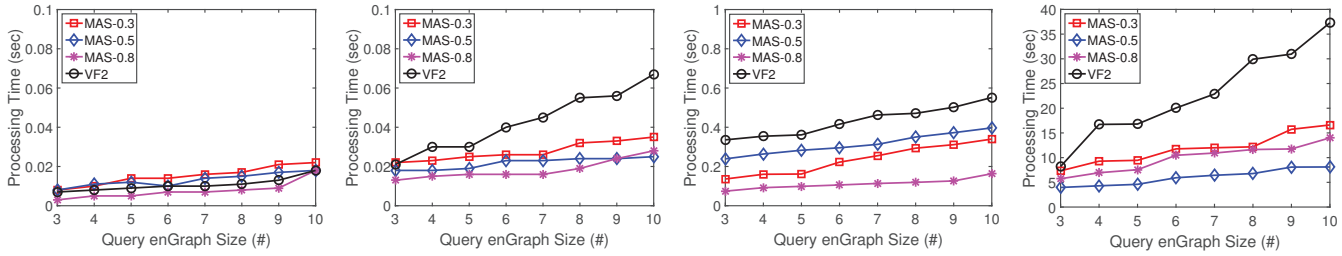
### 7.3.2 *The enGraph Efficiency*

We further tested the enGraph efficiency by measuring the generation time for data enGraph and query enGraph. The size of the data enGraph was ranged from 2 to 1000. The size of the query enGraph was ranged from 2 to 32 because the number of entities recognized by the state-




(a) Match accuracy @R32 (b) Match accuracy @R256 (c) Match accuracy @S2048 (d) Match accuracy @S8192

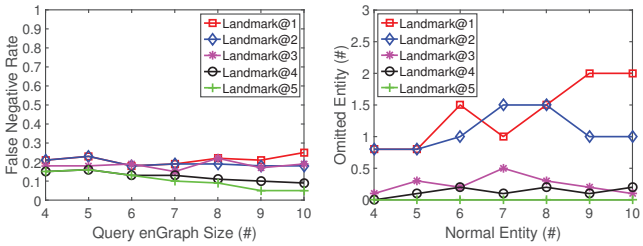
Fig. 11: Match accuracy produced by our MAS and the VF2 algorithms using the real and synthetic data.




(a) Processing time @R32 (b) Processing time @R256 (c) Processing time @S2048 (d) Processing time @S8192

Fig. 12: Processing time produced by our MAS and the VF2 algorithms using the real and synthetic data.

of-the-art entity recognition algorithms [19], [20] is up to 32 in our experiments. Specifically, the generation time for query enGraph includes the recognition time of entities and the building time of the query enGraph. Fig. 10 shows the time required to generate enGraphs for different numbers of entities. It can be seen from Fig. 10(a) that the data enGraph generation takes less than 0.85s for less than 1000 entities. Furthermore, it can be seen from Fig. 10(b) that the query enGraph generation takes less than 0.5s. Hence, the generation time for an enGraph demonstrates that the enGraph is conducive to improve the efficiency of indoor semantic floorplan updating.




(a) False negative rate (b) Omitted entitles

Fig. 13: The false negative rate and the number of omitted normal entities produced by SISE using the real data.

### 7.3.3 Performance of Subgraph Matching

The match accuracy and the query processing time of our matching algorithm were calculated to test the subgraph matching performance. In our experiments, the sizes of the data enGraph were 32, 256, 2048, and 8192, respectively. Since query enGraphs are usually small, their sizes are ranged from 3 to 10. Each experiment was run 10 times. In particular, for query processing, each run includes 100 query graphs. The average results were presented in this section. Furthermore, the VF2 algorithm [26] was considered as a baseline for comparison. Besides, the MAS algorithm was also evaluated under different similarities. That is, MAS-0.3 has similarity values larger than 0.3, MAS-0.5 has similarity values larger than 0.5, and MAS-0.8 has similarity values larger than 0.8.

It can be seen from Fig. 11 that MAS outperforms VF2 in all cases. That is because MAS is efficient to address the set similarity problem and uses an advanced quality score mechanism to return the correct matched subgraph. On one hand, the match accuracy of MAS and VF2 is increased, when the size of data enGraph is increased. On another hand, when the size of query enGraph is increased, the match accuracy of MAS and VF2 is decreased. The MAS-0.3 receives the best match accuracy.

It can be seen from Fig. 12 that MAS outperforms VF2 in most cases. Two common trends are observed: (1) When the data enGraph size is small, MAS processes queries as fast as VF2. (2) The influence of the data enGraph size for MAS is smaller than that for VF2. That is because VF2 does not compute the quality score of matched subgraphs as MAS does. When the query size is increased, the number of matched subgraphs is decreased, and VF2 cannot successfully filter the redundant search. MAS successfully limits the search space using the subspace constraints.

### 7.3.4 Detection Accuracy of Changed Entity

We further tested the detection accuracy of changed entities achieved by SISE on the real data. Let $N_l$ represent the number of landmarks. Let *Landmark*@1 represent 1 landmark, *Landmark*@2 represent $\frac{1}{4}N_l$ landmarks, *Landmark*@3 represent $\frac{3}{4}N_l$ landmarks, and *Landmark*@5 represent $N_l$-1 landmarks. Each experiment was run 10 times using different numbers of landmarks. The number of normal entities was ranged from 4 to 10 with a step of 1. First, the false

negative rate was measured using different numbers of landmarks. It can be seen from Fig. 13(a) that the false negative is decreased when the number of landmarks is increased. Second, the number of omitted entities caused by the false negative rate was measured. It can be seen from Fig. 13(b) that the number of omitted normal entities is decreased as the number of landmarks is increased. In summary, the above results have clearly demonstrated our theoretical analysis for Algorithm 2.

## 8 Discussion and Limitations

**Energy Consumption.** The energy consumption of the mobile application software was further tested, including the acquisition of images and inertial data and the communication of WiFi network. The PowerTutor profiler[5] was used to test the energy consumption in the Google Nexus 7 tablet. The energy consumed for inertial sensors (accelerometer, gyroscope, and geomagnetic) is about 30mW only. Capturing an image only consumes about $0.1$J. Furthermore, the energy consumption by WiFi network is $3.6$J for transmitting the inertial data and an image to the server. Compared to the battery capacity of 20k Joules, the collection and uploading of inertial sensor data and an image do not constitute any signification power consumption for a mobile device [4].

**Updating of General Entities in Different Same Planes.** Since it is difficult to extract the location relationship between two entities which are not in the same plane from an image, SISE can only update the changed entities lying on the same plane. Nevertheless, SISE can also update the entire indoor semantic floorplan by detecting multiple planes [42], [43]. In future, SISE will be extended to support multiple plane updating at the same time by combining other techniques, such as SfM technique [4], [10], [11].

**Updating of Annotated Objects.** Since some annotated objects (e.g., store names, posters) are tagged on the wall, it is difficult to unify them to the same plane and then update them using enGraphs in SISE. Moreover, compared to general entities, the key issue is how to accurately detect and recognize the text tagged in the annotated objects. Recently, several computer vision methods [44], [45] have been proposed to detect and recognize texts from images. In future, we will extend SISE to support annotated semantic updating.

## 9 Conclusion

In this paper, a self-updating method of the indoor semantic floorplan called SISE is proposed using a novel abstraction representation. First, two enGraph generation methods have been designed and a quality-based subgraph matching algorithm has been proposed to efficiently obtain the high-quality matched subgraphs. Two light-weight algorithms have then been proposed to accurately and efficiently detect the changed entities. These entities are then detected and their semantics in the indoor semantic floorplan are updated. Extensive experiments have been conducted on both real data and synthetic data. Experimental results demonstrate that SISE can effectively update $80\%$ out-of-date semantic of indoor general entities caused by indoor environmental variations.

5. http://ziyang.eecs.umich.edu/projects/powertutor/

## Acknowledgments

This work is partially supported by the National Natural Science Foundation for Outstanding Excellent young scholars of China under Grant No. $61422214$, National Natural Science Foundation of China under Grants No. $61772544$, $71471175$, $71690233$, $71331008$, and $1$, National Basic Research Program ($973$ program) under Grant No. $2014CB347800$, the Hunan Provincial Natural Science Fund for Distinguished Young Scholars under Grant No. $2016JJ1002$, the Guangxi Cooperative Innovation Center of cloud computing and Big Data under Grant Nos. $YD16507$ and $YD17X11$, and the National Postdoctoral Program for Innovative Talents under Grant No. $BX201600172$.

## References

[1] M. Alzantot and M. Youssef. Crowdinside: Automatic construction of indoor floorplans. In *Proc. of SIGSPATIAL*, Redondo Beach, California, 2012.

[2] Y. Jiang, X. Yun, X. Pan, K. Li, Q. Lv, R. P. Dick, L. Shang, and M. Hannigan. Hallway based automatic indoor floorplan construction using room fingerprints. In *Proc. of ACM UbiComp*, Zurich, Switzerland, 2013.

[3] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang. Walkie-Markie: Indoor pathway mapping made easy. In *Proc. of USENIX NSDI*, Lombard, IL, 2013.

[4] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *Proc. of ACM MobiCom*, Maui, Hawaii, 2014.

[5] D. Philipp, P. Baier, C. Dibak, F. Dürr, K. Rothermel, S. Becker, M. Peter, and D Fritsch. Mapgenie: Grammar-enhanced indoor map construction from crowd-sourced data. In *Proc. of IEEE PerCom*, Budapest, Hungary, 2014.

[6] S. Chen, M. Li, K. Ren, and C. Qiao. Crowdmap: Accurate reconstruction of indoor floor plans from crowdscourced sensor-rich videos. In *Proc. of IEEE ICDCS*, Ohio, USA, 2015.

[7] X. Guo, E. C. Chan, C. Liu, K. Wu, S. Liu, and L. M. Ni. Shop-profiler: Profiling shops with crowdsourcing data. In *Proc. of IEEE INFOCOM*, Toronto, Canada, 2014.

[8] M. Elhamshary and Y. Moustafa. Semsense: Automatic construction of semantic indoor floorplans. In *Proc. of IEEE IPIN*, Alberta, Canada, 2015.

[9] R. Meng, S. Shen, R. R. Choudhury, and S. Nelakuditi. Autolabel: Labeling places from pictures and websites. In *Proc. of ACM UbiComp*, Heidelberg, Germany, 2016.

[10] S. Y. Bao and S. Savarese. Semantic structure from motion. In *Proc. of IEEE CVPR*, Colorado Springs, CO, 2011.

[11] S. Y. Bao, M. Bagra, Y. Chao, and S. Savarese. Semantic structure from motion with points, regions, and objects. In *Proc. of IEEE CVPR*, Providence, RI, 2012.

[12] P. Jain, J. Manweiler, and R. R. Choudhury. Overlay: Practical mobile augmented reality. In *Proc. of ACM MobiSys*, Florence, Italy, 2015.

[13] M. Elhamshary, M. Youssef, A. Uchiyama, H. Yamaguchi, and T. Higashino. Transitlabel: A crowd-sensing system for automatic labeling of transit stations semantics. In *Proc. of ACM MobiSys*, Florence, Italy, 2016.

[14] H. Abdelnasser, R. Mohamed, A. Elgohary, M. F. Alzantot, H. Wang, S. Sen, R. Choudhury, and M. Youssef. SemanticSLAM: Using environment landmarks for unsupervised indoor localization. *IEEE Transactions on Mobile Computing*, 15(7):1770–1782, 2016.

[15] D. Yang, G. Xue, X. Fang, and J. Tang. Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *Proc. of ACM MobiCom*, Istanbul, Turkey, 2012.

[16] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao. TraviNavi: Self-deployable indoor navigation system. In *Proc. of ACM MobiCom*, Maui, Hawaii, 2014.

[17] Y. Shu, K. Shin, T. He, and J. Chen. Last-Mile navigation using smartphones. In *Proc. of ACM MobiCom*, Paris, France, 2015.

[18] S. Chen, M. Li, K. Ren, X. Fu, and C. Qiao. Rise of the indoor crowd: Reconstruction of building interior view via mobile crowdsourcing. In *Proc. of ACM SenSys*, Seoul, South Korea, 2015.

[19] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proc. of NIPS*, Quebec, Canada, 2015.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg. SSD: single shot multibox detector. In *Proc. of ECCV*, Amsterdam, 2016.

[21] S. Y. Bao, A. Furlan, F. Li, and S. Savarese. Understanding the 3D layout of a cluttered room from multiple images. In *Proc. of IEEE WACV*, Colorado Springs, CO, 2014.

[22] Y. Tian and J. M. Patel. TALE: A tool for approximate large graph matching. In *Proc. of ICDE*, Cancún, México, 2008.

[23] G. Gülsoy and T. Kahveci. RINQ: reference-based indexing for network queries. *Bioinformatics [ISMB/ECCB]*, 27(13):149–158, 2011.

[24] L. Hong, L. Zou, X. Lian, and P. Yu. Subgraph matching with set similarity in a large graph database. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2507–2521, 2015.

[25] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[26] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, 2004.

[27] S. Zhang, M. Hu, and J. Yang. TreePi: A novel graph indexing method. In *Proc. of ICDE*, Istanbul, Turkey, 2007.

[28] H. He and A. K. Singh. Closure-Tree: An index structure for graph queries. In *Proc. of ICDE*, Atlanta, GA, 2006.

[29] Y. Tian, R. C. McEachin, C. Santos, D. J. States, and J. M. Patel. SAGA: a subgraph matching tool for biological graphs. *Bioinformatics*, 23(2):232–239, 2007.

[30] M. Minot, S. Ndiaye, and C. Solnon. A comparison of decomposition methods for the maximum common subgraph problem. In *Proc. of IEEE ICTAI*, Vietri sul Mare, Italy, 2015.

[31] Q. Chen, Q. Sun, P. A. Heng, and D. Xia. A double-threshold image binarization method based on edge detector. *Pattern Recognition*, 41(4):1254–1267, 2008.

[32] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. In *Proc. of NIPS*, Barcelona, Spain, 2016.

[33] Z. Zhand. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[34] R. G. Sukthankar, R.and Stockton and M. D. Mullin. Smarter presentations: Exploiting homography in camera-projector systems. In *Proc. of ICCV*, British Columbia, Canada, 2001.

[35] P. Zhou, M. Li, and G. Shen. Use it free: Instantly knowing your phone attitude. In *Proc. of ACM MobiCom*, Maui, HI, 2014.

[36] X. Teng, D. Guo, Y. Guo, X. Zhou, Z. Ding, and Z. Liu. IONavi: An indoor-outdoor navigation service via mobile crowdsensing. *ACM Transactions on Sensor Networks*, 13(2):12:1–12:28, 2017.

[37] D. E. Knuth. *Estimating the efficiency of backtrack programs*. Technical report, Stanford, CA, USA, 1974.

[38] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of KDD*, Portland, Oregon, 1996.

[39] Y. Hu, X. Liu, S. Nath, and R. Govindan. ALPS: accurate landmark positioning at city scales. In *Proc. of ACM UbiComp*, Heidelberg, Germany, 2016.

[40] J. L. Schönberger and J. Frahm. Structure-from-motion revisited. In *Proc. of IEEE CVPR*, Las Vegas, NV, 2016.

[41] J. L. Schönberger, E. Zheng, M. Pollefeys, and J. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. of ECCV*, Amsterdam, The Netherlands, 2016.

[42] J. Piazzi and D. Prattichizzo. Plane detection with stereo images. In *Proc. of IEEE ICRA*, Orlando, Florida, 2006.

[43] D. F. Fouhey, D. Scharstein, and A. J. Briggs. Multiple plane detection in image pairs using J-linkage. In *Proc. of ICPR*, Istanbul, Turkey, 2010.

[44] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *arXiv:1406.2227*, 2014.

[45] S. Wang, F. Sanja, and R. Urtasun. Lost shopping! monocular localization in large indoor spaces. In *Proc. of IEEE ICCV*, Santiago, Chile, 2015.