A Graph-Based Topological Maps Generation Method for Indoor Localization

Zhixing Lin, Chundi Xiu, Wei Yang, Dongkai Yang
School of Electronic and Information Engineering
Beihang University
Beijing, China
Email: {linzhixing, xcd, yangwei89, edkyang}@buaa.edu.cn

Abstract—**Indoor maps are widely used to display user's location and refine pedestrian trajectories by enforcing constraints such as impassable walls. However, indoor maps are always unavailable due to their time-consuming and labor-intensive manual constructions. The widespread CAD drawings enable us to generate indoor maps at affordable costs. In this paper, we present a graph-based method for automatically generating topological indoor maps. We preprocess the initial data from CAD drawings and extract qualified door lines and wall lines. Rooms and corridor are extracted by detecting minimum cycle basis (MCB) of a walls-based graph. The indoor maps are then constructed by analyzing topology between all indoor spatial elements. In order to validate the generated maps, map matching algorithm using particle filter is implemented to calibrate the preliminarily estimated trajectories. Experiments results show that the proposed method runs much faster than previous work and the generated maps can significantly improve the average accuracy. Meanwhile, the number of cross-wall behaviors is also reduced.**

Keywords—indoor localization; indoor map; minimum cycle basis; map matching;

I. Introduction

With the increasing popularization of mobile devices, location information has gained its importance for ubiquitous computing and location-based service applications. Indoor map, which basically comprises three kinds of information (geometry, topology, and semantics), is a set of geographic data showing structures, points of interest and pathways inside a building. As the critical part of indoor localization system, indoor map can be used to display users' location and provide navigation services. Despite that intensive researches on location estimation have been conducted [1], the shortage of indoor maps has limited the availability and spread of indoor localization applications. Unlike outdoor environment where several commercial maps with large-scale urban road network are already existed such as Google Map, Baidu Map and so on, there is still a lack of large-scale maps for indoor use. Some related companies such as Google, AutoNavi, and Palmap construct indoor maps by surveying on-site and acquiring indoor spatial data manually, which is obviously a time-consuming and labor-intensive process. In addition, map matching can fuse the preliminarily estimated trajectories with a topological indoor map to improve location results, e.g. [2-3] utilize walls of indoor environment to restrain pedestrian's movement based on a simple principle that pedestrian cannot pass through walls and can only enter or leave a room through a door. In their studies, particle filters are implemented by combining relative movement measurements with environmental constraints to reduce location estimation errors. Nevertheless, since the required indoor maps are often unavailable, researchers construct maps manually for some specific experimental scenarios, which is impractical to be carried out in real scenarios considering for its huge costs. Therefore, a practical solution for generating feasible topological maps for indoor localization system within affordable costs is in a great need.

Simultaneously Localization and Mapping (SLAM) is used for a robot to construct maps of unknown environments while simultaneously tracking itself [4]. It only needs a few manual operations before a highly accurate map is generated. However, it suffers from the disadvantage that the robot and its facilities (odometry, laser range sensors, etc.) are rather expensive to afford. Moreover, the robot has to discover all building entities of the indoor environment to generate a complete map. [5] converted CAD drawings to CityGML models [6]. Unfortunately, it requires a large number of user interactions and has to meet several restrict prerequisites such as all rooms shall be depicted as closed polygons. [7] proposed a parser for generating topological indoor maps from CAD drawings. It consists of three stages including preprocessing, extraction and post processing. To extract rooms, the parser traverses all lines and grows a polyline using depth-first search until the polyline forms a closed polygon as a room candidate. Unqualified and duplicated rooms are removed from the room candidates according to some post processing rules. This parser can achieve satisfying results from those simple CAD drawings with high quality and small scale. However, its computation complexity and runtime would be increased notably when CAD drawings are complicate with a bigger scale and a larger number of lines, which is not appropriate for generating maps in many actual scenarios. Considering the shortcomings above, [8] proposed a pruning algorithm during the polyline-growing process. A maximal number K is set at the beginning. Once the number of polyline-growing is greater than K, the growing process is then stopped. Results reveal that the proposed pruning algorithm can decrease the time complexity to some extent. But it has to adjust the number K manually in different scenarios. Other researchers also leverage sensors in smartphones to estimate indoor floorplan via mobile crowdsensing [9], but the estimated floorplans are still highly error-prone and immature for location display in reality.

Aiming at providing a practical and affordable solution to feasible topological maps generation for indoor localization, we present a method for extracting indoor spatial elements from the preprocessed data of CAD drawings by detecting minimum cycle basis of a walls-based graph in this paper. The proposed method owns an advantage that its time complexity is considerably smaller than previous work. In addition, map matching algorithm using particle filter is investigated to validate the generated map. In summary, we make the following contributions in this paper:

- The system architecture for generating topological indoor maps from CAD drawings is given.
- A novel algorithm for extracting indoor spatial elements based on minimum cycle basis of a walls-based graph is proposed.
- Experiments are conducted in a campus building to validate that the generated indoor map can be fused with a preliminarily estimated trajectory to improve the location results.

The rest of this paper is organized as follows: Section II describes the map generation system in detail. The performance analysis and experiments results are shown in Section III. Section IV concludes this paper with suggestions for future work.

II. System Design

A. System Overview

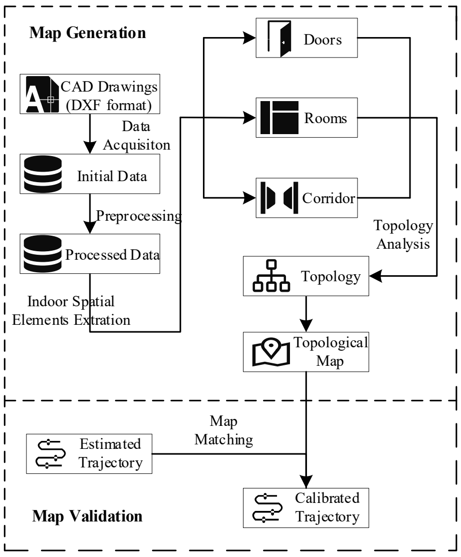

Fig. 1. System architecture

The whole system consists of two main subsystems, i.e. Map Generation and Map Validation. As shown in Fig.1, Map Generation is responsible for generating indoor maps from CAD drawings. It comprises several steps, namely, data acquisition and preprocessing, indoor spatial elements extraction, and topology analysis. Map Validation implements the map matching algorithm to improve preliminarily estimated trajectories for localization.

B. Data Acquisition and Preprocessing

Buildings in real scenario generally consist of various indoor spatial elements such as doors, walls, stairs, elevators, rooms, corridor, even chairs and desks, etc. In architectural design, architects often illustrate CAD drawings and use low-level geometries including lines, arcs, and polylines to depict the internal structure of buildings in detail.

Besides, CAD drawings depicting indoor environments often comprise data in multiple categories. Namely, geometric lines, arcs, and blocks are used to depict walls, doors and columns, respectively. Similarly, layer WALL, DOOR, and COLUMN are used to organize and distinguish all geometric data. Since indoor map can be illustrated simply by basic elements such as walls and doors, we then remove redundant and trivial elements such as columns and fire hydrants from the original CAD drawing by a few user interactions in the early stage and choose essential layers that depict a complete internal structure.

AutoCAD Drawing Exchange Format (DXF) is a CAD data file format developed by Autodesk for data interchange between AutoCAD and other software [10]. In fact, we acquire the initial building entities' geometric data by analyzing CAD drawings in DXF format in this paper.

However, owing to the lack of a unified standard on how CAD should be illustrated as well as the difference of illustrators, precision errors often appear if the illustrator is not careful enough when creating the drawing. What's more, there are always expression differences between CAD drawings and indoor maps. For instance, walls often consist of two parallel lines indicating the thickness of the wall in CAD drawings, while we usually illustrate walls in one single line in indoor maps. For these reasons, initial data has to be preprocessed before indoor spatial elements are extracted.

To solve the above mentioned problems of error, redundancy and expression difference in CAD drawings, we fix errors and simplify the initial data using error tolerance calculation, which is a preprocessing actually. A point defined as a two-dimension coordinate vector $p \in \mathbb{R}^2$ is the basic data of indoor map database. A line is defined as a set of two endpoints $\mathbb{L} = \{(p,q) \mid p,q \in \mathbb{R}^2\}$. On this basis, some definitions of geometric relations are given below:

a) Point-Point Approximation: point p and point q are *approximate* iff their Euclidian distance is smaller than a predefined threshold ϵ_1.

$$p \approx q \Leftrightarrow dis(p,q) < \epsilon_1 \tag{1}$$

b) Line-Line Approximation: line L_1 and line L_2 are *approximate* iff their endpoints are approximate.

$$L_1 \approx L_2 \Leftrightarrow \{\forall p \in L_1,\ \exists q \in L_2 \mid p \approx q\} \tag{2}$$

c) Point-Line Adjacency: point p is *adjacent* to line L iff p and one of L's endpoints are approximate.

$$p \xleftrightarrow{adj} L \Leftrightarrow \{\exists q \in L \mid p \approx q\} \tag{3}$$

d) Line-Line Adjacency: line L_1 is *adjacent* to line L_2 iff one of L_1's endpoints is adjacent to L_2.

$$L_1 \xleftrightarrow{adj} L_2 \Leftrightarrow \left\{\exists p \in L_1 \mid p \xleftrightarrow{adj} L_2\right\} \tag{4}$$

e) Line-Line Parallel: line L_1 and line L_2 are *in parallel* iff the dot product of two line vectors approximately equals the product of their lengths.

$$\begin{gathered} L_1 \parallel L_2 \Leftrightarrow \\ \left|\left|(p_2 - p_1) \cdot (q_2 - q_1)\right| - dis(p_1, q_1) \cdot dis(p_2, q_2)\right| < \epsilon_2 \end{gathered} \tag{5}$$

f) Distance between Two Parallel Lines: Given two parallel lines L_m and L_n, the distance between them $dis(L_m, L_n)$ are defined as d_1 (the distance from point q_1 to line L_2) and d_2 (the minimum distance from endpoints of L_3 to endpoints of L_4) according to two different circumstances below in Fig.2:

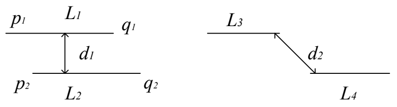

Fig. 2. Two forms of parallel lines: (a) there is an overlap between projections; (b) there is no overlap between projections

g) Line-Line Vicinity: parallel lines L_1 and L_2 are *in vicinity* iff their distance is smaller than a threshold ϵ_3.

$$L_1 \xleftrightarrow{near} L_2 \Leftrightarrow dis(L_1, L_2) < \epsilon_3 \tag{6}$$

Based on relations defined above, we preprocess the initial data following rules below:

- Remove lines whose lengths are extremely small (isolated points included): if endpoints p and q of line L are approximate, $p \approx q$, then L shall be removed.
- Remove approximate lines: if line L_1 and line L_2 are approximate, $L_1 \approx L_2$, then one of them shall be removed.
- Fix erroneous lines: if line L_1 is adjacent to line L_2, $L_1 \xleftrightarrow{adj} L_2$, then update their endpoints' coordinates to the approximated coordinates.
- Extract walls' midlines: if line L_1 and L_2 are in vicinity, $L_1 \xleftrightarrow{near} L_2$, then L_1 and L_2 are walls illustrated by two parallel lines. We then extract their midlines and convert the double-lines-illustrated walls to single-line-illustrated walls, which are more commonly used in indoor maps. The wall midlines should not only keep their original connections but also divide the indoor space correctly. Fig.3. shows different scenarios of midlines extraction. Note that the extracted midlines are viewed as the wall lines in the following part of this paper.

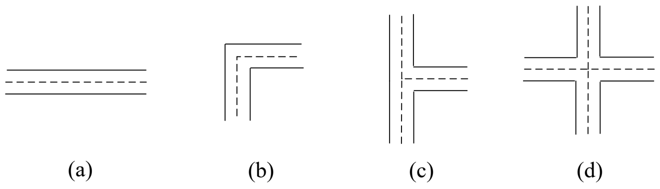

Fig. 3. Different scenarios of walls' midlines extraction. The dotted lines are the extracted midlines.

C. Doors Extraction

In a regular CAD drawing, doors are typically organized in the specific layer DOOR and geometrically depicted by two types of arcs, namely, single arc and double arcs. The dotted lines shown in Fig.4 indicate the door sills. Since the doors depicted by arcs are not suitable for the expression, organization and topology analysis of indoor maps as well as the map matching, we deem door sills depicted by lines as door lines instead. For those doors depicted by single arc, we build two lines starting from the arc center to its two endpoints. The line that is parallel to wall lines is then chosen as door line. For those doors depicted by double arcs, they can be viewed as two doors depicted by a single arc, and two door lines can be extracted following the instruction above. We then merge those two short door lines as one long door line. The whole process of doors extraction is described in Algorithm 1.

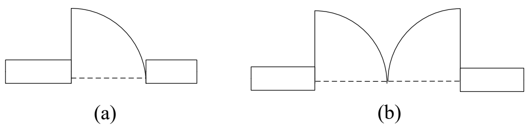

Fig. 4. Two forms of door in CAD: (a) single arc; (b) double arcs;

D. Rooms Extraction

Generally, rooms are not provided directly by CAD drawings, but are depicted by geometric lines in layers such as WALL, COLUMN, and DOOR. Note that, unlike the method proposed in [7], we do not analyze stairs and elevators separately, but treat them as rooms with specific functions considering that they have the same basic structure with normal rooms. Apparently, it would not affect the indoor spatial data queries, location display, and map matching algorithm in the end. Thus, we extract stairs and elevators using the same algorithm for extracting rooms.

Since we already preprocessed the initial data of CAD drawings and extracted the door lines and wall lines (walls' midlines) before, we view the interconnected door lines and wall lines as a set of intersected lines. The room extraction is considered to be polygons detection from the lines set, which can also be considered to cycle detection in graph theory. Methods proposed in [7,8] detected cycles by growing polylines continuously inspired by depth-first search in computer science. Nonetheless, the runtime using these methods is fairly long. Besides, there are always duplicate

Algorithm 1 doorsExtraction

Input: CAD model of indoor space in DXF format: DXF

Output: a set of door lines: $doors$

```
$doors = \text{doorsExtraction}(DXF)$
1  $doors \leftarrow \Phi$
2  /*acquire data from the $DOOR$ layer*/
3  $doorLayer \leftarrow \text{layer}(DXF, \text{'}DOOR\text{'})$
4  for each $arc$ in $ARCS(doorLayer)$ do
5      $newLine1 = \text{Line}(arc.center, arc.startpoint)$
6      $newLine2 = \text{Line}(arc.center, arc.endpoint)$
7      if $(newLine1 \parallel wall)$
8          add $newLine1$ to $doors$
9      else
10         add $newLine2$ to $doors$
11     end
12 end for
13 /*merge doors depicted by double arcs*/
14 for each $line1, line2$ in $LINES(doors)$ do
14     if $\left(line1 \overset{adj}{\longleftrightarrow} line2\right)$
15         $line3 = \text{Line}(line1.startpoint, line2.startpoint)$
16         add $line3$ to $doors$
17         remove $line1, line2$
18     end
19 end for
20 return $doors$
```

polygons and incorrect polygons comprising some smaller polygons inside those extracted polygons, which means a series of post processing is still in need. In fact, rooms, as physical entities, shall be considered as the minimum polygons which cannot be composed of other polygons, corresponding to the *minimum cycle basis(MCB)* in graph theory [11,12].

1) *MCB Introduction*

Given a graph $G = (V, E)$, let X and Y be two sets of edges of G, following operations are defined:

$$\begin{cases} X \oplus Y = (X \cup Y) - (X \cap Y) \\ 1 \cdot X = X,\ 0 \cdot X = \emptyset \end{cases} \tag{7}$$

then *cycle space* of G is generated in $GF(2)$ [13]. The dimension of cycle space is given by the *cyclomatic number* $\nu = |E| - |V| + P$, where P is the number of connected components of G [14].

To each cycle C in a graph $G = (V, E)$, an incidence vector x with length $|E|$ is defined, where $x_e = 1$ if Edge e belongs to C. Otherwise, $x_e = 0$. Given any collection of cycles, it is called *independent* if their incidence vectors are independent over $GF(2)$. A maximal independent collection of cycles is called a *cycle basis*.

Consider a graph G with edge weights. The *weight of a cycle* is the sum of its edges, and the *weight of a collection of cycle* is the sum of the weights of its cycles. A *minimum cycle basis (MCB)* of G is a cycle basis of minimum weight.

2) *MCB Extraction*

We extract minimum polygons directly by detecting the MCB of a graph based on the preprocessed data from CAD drawing.

Given the wall lines and door lines extracted before, a graph G is then represented, where the nodes V are the endpoints of all lines, the edges E are all lines with weights w of their lengths. G can be divided into P connected components $Comp_i$ $(i = 1, 2, \cdots, P)$ according to its connectivity, where P is the number of connected components. For every connected component $Comp_i$, the shortest path between any nodes pair Π_{xy} in graph G then is calculated. Given any node v in graph G, if there is an edge $E(x,y)$ that enables the shortest path Π_{xv} from node x to node v and the shortest path Π_{vy} from node v to node y to share the same node v, the cycle $C = \Pi_{xv} \cup \Pi_{vy} \cup E(x,y)$ is added to cycle candidates $Cycles_i$ of $Comp_i$. We then sort all cycles C_j $(j = 1, 2, \cdots, N_i)$ of cycle candidates $Cycles_i$ by ascending length in different connected components, where N_i is the number of cycles in $Cycles_i$. After that, a 0-1 incidence matrix with size $N_i \times |E|$, whose rows represent different cycles and columns represent the edges of the connected component, is defined. Gaussian elimination using elementary row operations over the integers modulo two can then be applied to the incidence matrix, processing each row in turn, until all independent vectors are found, i.e. the MCB is found. Finally, we extract the minimum polygons through the edges and nodes of MCB.

E. Corridor Extraction

To some extent, corridor is similar to room from the point view of architecture design as both of them consist of a series wall lines and door lines. However, corridor is still different from a room since corridor is typically illustrated by a nested structure consisting of one outer cycle and one or more inner cycles, rather than illustrated by a simple polygon. A typical corridor is shown in the shaded area of Fig.5. The corridor is composed of cycles C_1, C_2 and C_3 altogether, where C_1 is the outer cycle and C_2, C_3 are the inner cycles respectively. Thus, we cannot extract corridors directly by detecting MCB of a graph. Instead, we have to extract it using an indirect method.

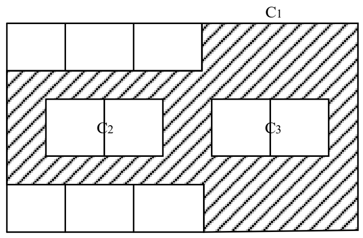

Fig. 5. An illustration of corridor

```
**Algorithm 2** roomsAndCorridorExtraction
**Input:** the graph representd by wall lines and door lines: $G$
**Output:** a set of rooms: $rooms$
         parts of corridor: $outerCor, innerCor$

$(rooms, outerCor, innerCor) =$ roomsAndCorridorExtraction$(G)$
1  $rooms \leftarrow \Phi$
2  $COMPS =$ getAllConnectedComp$(G)$
3  for each $comp$ in $COMPS(G)$ do
4      $circles \leftarrow \Phi$
5      /* calculate all pairs of shortest paths */
6      $\Pi =$ getAllPairsShortestPaths$(comp)$
7      for each $v$ in $VERTEX(comp)$ do
8          for each $E(x,y)$ in $EDGES(comp)$ do
9              if $(\Pi_{xv} \cap \Pi_{vy} = v)$
10                 $C = \Pi_{xv} \cup \Pi_{vy} \cup E(x,y)$ //find a cycle candidate
11                 add $C$ to $circles$
12             end
13         end for
14     end for
15     $circles =$ sortByLength$(circles)$
16     $matrix =$ getMatrix$(circles)$ //define an incidence matrix
17     /* find all independent vectors */
18     $vectorLinearlyIndept =$ gaussianEliminationMod2$(matrix)$
19     /* find MCB */
20     $minCircles =$ getCircles$(vectorLinearlyIndept)$
21     /* find the maximal cycle */
22     $maxCircle =$ getCircles(sumMod2$(vectorLinearlyIndept)$)
23     for each $circle$ in $minCircles$
24         for each $E(x,y)$ in $EDGES(circle)$
25             $room.VERTEX = x,y$
26             $room.EDGES = E(x,y)$
27         end for
28     add $room$ to $rooms$
29     end for
30 end for
31 $outerCor =$ max(sortByLength(all $COMPS.maxcircle$))
32 $innerCor =$ other $COMPS.maxcircle$
33 return $(rooms, outerCor, innerCor)$
```

For every connected component, we sum all the independent vectors of incidence matrix found in Subsection *D 2)* over the integers modulo two and get a maximal cycle. All weights of maximal cycles are also calculated. Then the cycle with the maximal weight is determined to be the outer cycle, while the rest cycles are the inner cycles. The whole process of rooms and corridor extraction is described in Algorithm 2.

F. Topology Analysis

When indoor spatial elements (doors, rooms and corridor) are extracted successively, we integrate them into an indoor spatial model and analyze the topology of all those spatial elements, which actually are relations between the entire indoor space and all spatial elements as well as relations between spatial elements themselves. Namely, an indoor map in one specific floor of a building often includes several rooms (stairs and elevators are viewed as a special room as we already discussed before) and a corridor. Likewise, a room or a corridor is composed of a plenty of walls and doors. In addition, some rooms are often connected with other rooms or corridor through doors. We then build an object-oriented topological structure based on these relations.

Fig.6 shows the generated topological structure, where *IndoorMap* represents the indoor map in one specific floor of a building, *IndoorMap* comprises several rooms *Room* and a corridor *Corridor*. *Room* and *Corridor* are composed of multiple walls *Wall* and doors *Door*. Endpoints indicating the coordinates information define the *Wall* and *Door*, which are the basic elements generating the topological map. Two rooms are considered to be *AdjRoom* when they share one door or more. Similarly, rooms and corridor are considered to be *AdjCorridor* when they share doors too. The function *GetMap* is defined to determine the corresponding indoor map *IndoorMap* once a *Room* is known. The function *GetRoom* is also defined to determine the corresponding Room once a *Wall* and a *Door* are known.

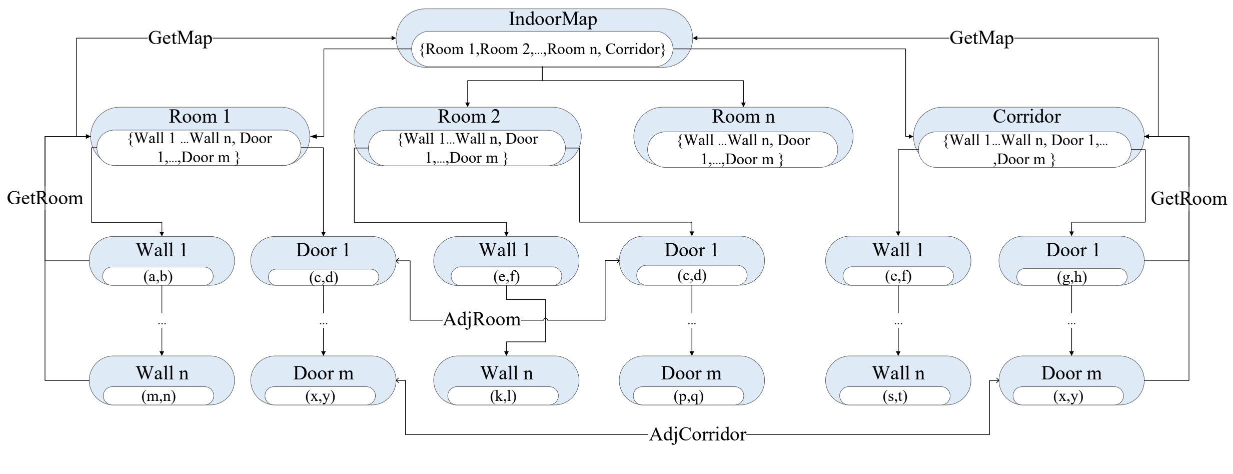

Fig. 6. Topological structure of indoor map

III. Analysis and Experiments

A. Performance Analysis

The performance of methods for map generation mainly depends on the detection quality of indoor spatial elements and the runtime of the whole process. In terms of the detection quality, it relies on the preprocessing of initial data from CAD drawing. In this respect, the proposed method does not make differences essentially from previous work. Actually, the correct detection ratio of doors, rooms, and corridor in our work are same 100% with previous work in all following experiments using a few user interventions. Hence, we focus on analyzing runtime or time complexity in this section.

Since rooms and corridor are the major components of indoor spatial elements, their extraction should be deemed a critical step of the proposed method. We evaluate the performance of rooms and corridor extraction by detecting MCB and compare it with methods proposed in [7,8]. We name the algorithms proposed in [7] and [8] as Depth First Search (DFS) and Kth Depth First Search (KDFS), respectively. Our algorithm is then named as Minimum Cycle Basis Detection (MCBD) correspondingly.

Analyzing the whole process in Algorithm 2, we know that the loop from step 3 to step 30 repeats P times since the graph consists of P different connected components. Therefore, we consider their time complexity in total. The shortest paths calculation in step 6 can be performed in $O(|V|^3)$ time using Floyd algorithm altogether [15]. Likewise, the cycle candidates from step 7 to step 14 takes a time complexity in $O(|E||V|)$ in all. Assume that there are N cycles of P connected components found during the previous steps, then the sorting step can be performed in $O(N\log N)$ time with an upper bound $O(\nu|V|\log(|V|))$ using a quick-sort algorithm. The Gaussian elimination step dominates the time complexity from other steps with a $O(|E|\upsilon^2|V|)$ time complexity. Considering the architecture in real scenarios, the graph G represented in the first step shall be a planar graph. Hence, $O(|E|\upsilon^2|V|)=O(|E|^3|V|)=O(|E|^4)$, we then can conclude that the MCBD algorithm has a worst upper bound $O(|E|^4)$.

By comparison, the time complexity of DFS has a upper bound $O(|E|\cdot M^{|E|})$, where M is the maximal number of lines that a line in the collection of wall lines and door lines adjacent to. Similarly, the time complexity of KDFS has a upper bound $O(|E|\cdot M^K)$, where K is the number of polyline-growing. Obviously, the time complexity of both DFS and KDFS are exponential, while MCBD is a polynomial-time algorithm. Undoubtedly, MCBD runs much faster than those two previous methods when the number of edges increases.

B. Map Generation

To evaluate their performance in real scenarios, we implement the algorithms in MATLAB and tested in an Intel Core i3 3100M, 2.40GHz, 10G RAM laptop running Windows 10 (64bit version). The tested files are CAD drawings in five scenarios consisting of a different number of lines. The runtime of three algorithms is listed in Table I and a comparison of different algorithms' runtime is shown in Fig.7. Note that the number of polyline-growing K of KDFS in different scenarios are set to the minimum numbers that can achieve same correct detection ratio with DFS and MCBD.

As we can see in Fig.7, the MCBD algorithm proposed in this paper has a much smaller runtime than those two previous algorithms. One thing deserved to point out is that Scenario IV and Scenario V are actually at the 7th floor and 6th floor in F Block of New Main Building in Beihang University with approximately 40m*35m area and 40m*115m area respectively. Moreover, Scenario V is much more complicated and has a considerably larger number of lines than the other four scenarios. In Scenario V, the runtime of DFS and KDFS is noticeably longer (more than 24 hours) than the runtime of MCBD, which accords with the performance analysis above exactly. The indoor spatial database of 6th floor in F Block of New Main Building is presented as an example in Table II, where the numeral indicates the serial number of the walls lines and door lines. Two generated maps of 7th floor and 6th floor are shown in Fig.8. Rooms are divided into accessible or inaccessible and painted in two different colors if there is a door inside or not in this case. In fact, one can paint each room in different color if needed.

TABLE I. Comparison of Different Algorithms' Runtime (Units: Seconds)

Scenarios	Number of Lines	DFS	KDFS	MCBD
I	39	42.21	16.92	11.04
II	48	247.16	82.85	20.14
III	58	1273.42	228.21	23.39
IV	142	5673.24	996.70	141.94
V	301	>86400	>86400	654.48

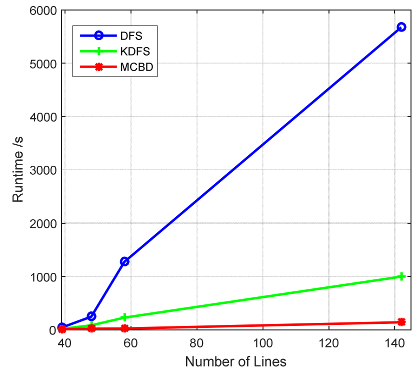

Fig. 7. Comparison of different algorithms' runtime

TABLE II. INDOOR SPATIAL DATABASE OF 6TH FLOOR IN F BLOCK OF NEW MAIN BUILDING

6th Floor in F Block of New Main Building							
Room 1		…		Room 46		Corridor	
Wall	Door	Wall	Door	Wall	Door	Wall	Door
(1,2)	(3,4)	…	…	(226,228)	(233,234)	(5,7)	(3,4)
…		…		…		…	…
(7,8)		…		(300,301)		(240,241)	(233,234)

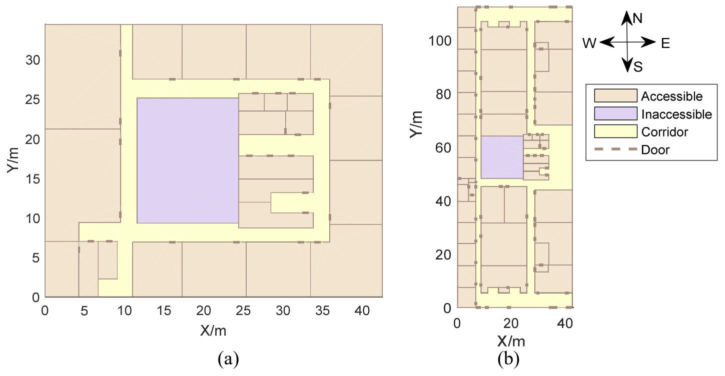

Fig. 8. Generated maps of F Block of New Main Building: (a)7th floor; (b) 6th floor;

C. Map Validation

Map matching is used to tackle the drift problems in inertial measurement units (IMUs). To validate the generated topological maps, we implement the map matching algorithm using particle filter [16]. Location results estimated by Pedestrian Dead Reckoning (PDR) [17] are viewed as the measurements. The generated maps provide constraints such as impassable walls to calculate the posterior probability.

Similar to the algorithm proposed in [2], there are typically three stages in map matching algorithm using particle filter: re-sampling, propagation, and correction. To utilize the generated topological map, the room number of particles are also defined besides their weights when all particles are initialized. A particle's initial room number is determined by its initial location. Thus, the adapted particle filter is depicted below.

A particle is propagated from the step $k-1$ to the step k with a step length and heading information in the step vector. To update the probabilities, we set the weight to zero if its step vector intercepts a wall line of the room where the particle was located. Otherwise, we then determine whether its step vector intercepts a door line of the room. If a cross-door behavior is detected, we keep its weight and set its room number to the new room where it is located after propagation according to the room connections or topology. Except for those two circumstances above, the particle keeps its weight and room number. The whole process of particle propagation and correction is described in Algorithm 3.

```
Algorithm 3 particlesUpdate
Input: particle in step $(k-1)$: $P_{k-1}$
       topological map: $tMap$
Output: particle in step $k$: $P_k$
$P_k = \text{particlesUpdate}(P_{k-1}, tMap)$
1  for each $P_k^i$ in $P_k$ do
2      /* particle is propagated from step k − 1 to step k */
3      $P_k^i \leftarrow P_{k-1}^i$
4      /* if step vector cross a wall */
5      if $(\text{step}(P_{k-1}^i, P_k^i)$ cross $walls$ in $tMap.rooms(P_{k-1}^i.roomNum))$
6          $P_k^i.w = 0$
7      else
8      /* if step vector cross a door */
9          if $(step(P_{k-1}^i, P_k^i)$ cross $doors$ in $tMap.rooms(P_{k-1}^i.roomNum))$
10             /* update the particle's room number */
11             $P_k^i.rNum = \text{updateRoomNum}(P_{k-1}^i.roomNum)$
12         end
13     end
14 end for
15 return $P_k$
```

In the experiments, we collect data from sensors (accelerometer, gyroscope, and magnetometer) built in an Android smartphone (Smartisan M1, Android 6.0.1) at the 7th floor (Experiment 1) and 6th floor (Experiment 2) in F Block of New Main Building where the two generated topological maps belong to. After the implementation of the PDR algorithm, the pedestrian trajectories are estimated and displayed on the generated maps. The total moving distance of two trajectories are about 80 and 356 meters, respectively.

The trajectories are shown in Fig.9. As we can see, the preliminarily estimated trajectories suffer from an increasing drift over time due to accumulative errors of inertial sensors and cross walls plenty of times in the cross wall points (CWP), dividing each trajectory into several normal segments and cross-wall segments (CWS) which are some parts of trajectories that start from a wall and end until a passable path is found. Clearly, the improved trajectories are much closer to the ground truth and also have decreased cross-wall behaviors after the map matching. To evaluate the impact of map matching more precisely, the Percentage of Cross-Wall Distance (PCWD) is defined as follow.

$$PCWD = \frac{\sum \text{distance of cross} - \text{wall segment}}{\text{total distance of trajectory}} \quad (8)$$

Obviously, a ground truth has a PCWD of 0 and an estimated trajectory gets a smaller PCWD if it is closer to the ground truth. As listed in Table III, the number of cross-wall behaviors in two experiments is reduced from 10 to 2 and from 37 to 8 respectively. The PCWD is also reduced from 32.7% to 7.2% and from 63.8% to 5.6% with their average accuracy of 0.74m and 1.92 m after map matching. A 90th percentage of the Euclidean distance errors between the true location and the estimated location is less than 1.5m and 3.9m as shown in the cumulative distribution in Fig.10.

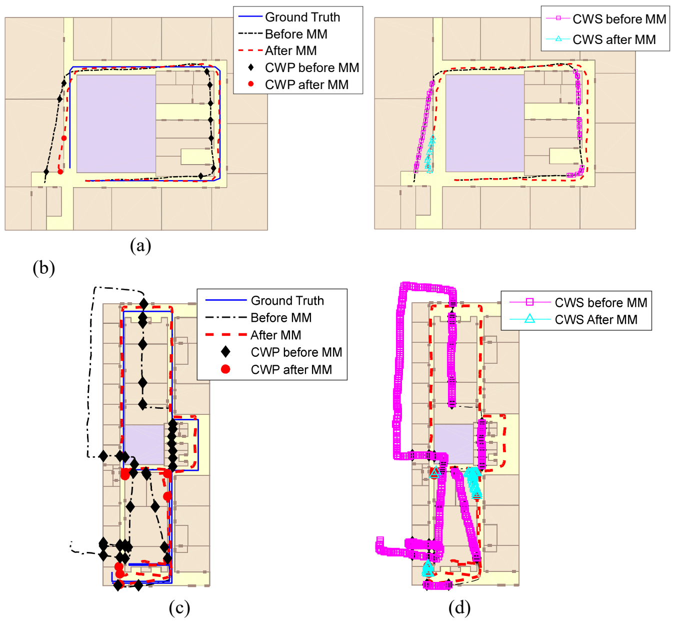

Fig. 9. Experimental trajectories in 7th floor and 6th floor before and after map matching (MM), and ground truths as well as: (a) (c) cross-wall points (CWP); (b) (d) cross-wall segments (CWS);

TABLE III. COMPARISON OF EXPERIMENTAL RESULTS BEFORE AND AFTER MAP MATCHING

Scenarios	**Map matching**	**Number of cross-wall points**	**Percentage of cross-wall distance (PCWD)**	**Average accuracy**
Exp.1	Before MM	10	32.7%	1.82 m
	After MM	2	7.2%	0.74 m
Exp.2	Before MM	37	63.8%	12.70 m
	After MM	8	5.6%	1.92 m

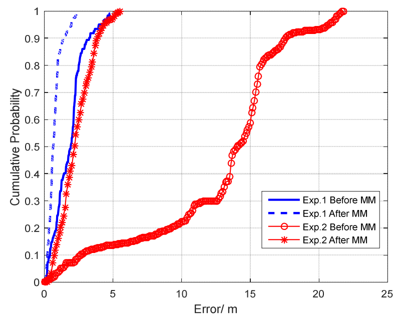

Fig. 10. Cumulative distribution function results

IV. CONCLUSION

Indoor map is a foundation of indoor localization system. Apart from displaying location results and providing navigation services, indoor map can also contribute to the map matching by enforcing walls constraints. In this paper, we present a topological indoor maps generation method from CAD drawings. A complete system for generating indoor maps is described, which is basically composed of data preprocessing, spatial elements extraction and topology analysis. An effective and fast algorithm for extracting rooms and corridor by detecting minimum cycle basis of a walls-based graph is also proposed. We implemented our map generation method in several scenarios and validated the generated maps via map matching. Analysis and experiments reveal that the proposed method can generate indoor maps effectively and calibrate the location estimation errors significantly. Since we only discussed geometry and topology of indoor maps in this paper, future works might include the supplement of semantics as well as the integration of indoor spatial database management system.

ACKNOWLEDGMENT

This work is supported by the Beihang BeiDou Technology Industrialization Funding Program (Grant No. BARI1701).

REFERENCES

[1] Shang Jianga, et al. “Improvement schemes for indoor mobile location estimation: A survey.” *Mathematical Problems in Engineering* 2015 (2015).

[2] Woodman Oliver, and Robert Harle. “Pedestrian localisation for indoor environments.” *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008.

[3] Perttula Arto, et al. “Distributed indoor positioning system with inertial measurements and map matching.” *IEEE Transactions on Instrumentation and Measurement* 63.11 (2014): 2682-2695.

[4] Durrant-Whyte Hugh, and Tim Bailey. “Simultaneous localization and mapping: part I.” *IEEE robotics & automation magazine* 13.2 (2006): 99-110.

[5] Groneman A., and S. Zlatanova. “Toposcopy: A modelling tool for citygml.” *Gsdi Association* (2009).

[6] Gröger Gerhard, et al. “OpenGIS city geography markup language (CityGML) encoding standard.” *Open Geospatial Consortium Inc* (2008): 1-234.

[7] Schäfer Martin, Christian Knapp, and Samarjit Chakraborty. “Automatic generation of topological indoor maps for real-time map-based localization and tracking.” *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*. IEEE, 2011.

[8] Xu Dazhou. “Research on data acquisition and generation for indoor space. ” Master dissertation, University of Science and Technology of China, 2016

[9] Alzantot Moustafa, and Moustafa Youssef. “Crowdinside: automatic construction of indoor floorplans.” *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012.

[10] Autodesk *DXF Reference*. 2012.https://www.autodesk.com/techpubs/ autocad/acadr14/ dxf/dxf_reference.htm

[11] Horton Joseph Douglas. “A polynomial-time algorithm to find the shortest cycle basis of a graph.” *SIAM Journal on Computing* 16.2 (1987): 358-366.

[12] Hartvigsen David, and Russell Mardon. “The all-pairs min cut problem and the minimum cycle basis problem on planar graphs." *SIAM Journal on Discrete Mathematics* 7.3 (1994): 403-418.

[13] Bondy John Adrian, and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*. Vol. 290. London: Macmillan, 1976.

[14] McCabe Thomas J. “A complexity measure.” *IEEE Transactions on software Engineering* 4 (1976): 308-320.

[15] Cormen Thomas H. *Introduction to algorithms*. MIT press, 2009.

[16] Hightower Jeffrey and Gaetano Borriello. “Particle filters for location estimation in ubiquitous computing: A case study.” *UbiComp 2004: Ubiquitous Computing* (2004): 88-106.

[17] Harle Robert. “A survey of indoor inertial positioning systems for pedestrians.” *IEEE Communications Surveys and Tutorials* 15.3 (2013): 1281-129